HEWLETT PACKARD

HP-71

Users' Library Solutions

Math

HEWLETT PACKARD

USERS' LIBRARY ORDER FORM

HP-71 Solutions Books are available at a cost of $12.50 each. To order, please call toll free (800) 538-8787, contact your local dealer, or use the form provided below to order directly from the Users' Library. If ordering from the Users' Library, please include a handling charge of $3.50 per order.

REMEMBER...

You can further customize your HP-71 with software available through the Users' Library.

A Users' Library Membership includes 2 free Solutions Books (a $25.00 value) plus periodic updates containing:

- –Product Information
- –Contests
- –Discounts
- –New Software Availability
- –Third-Party Products

YES! I want to subscribe to the Users' Library.

☐ $20.00 U.S. and Canada (5851-1011)
(plus state and local taxes)

☐ $35.00 OUTSIDE the U.S. and Canada (5851-1012)

Solution Books Requested

1. ______________________________
Title Part Number

2. ______________________________
Title Part Number

HP-71 Solutions Book Selection includes:

00071-90064/Math
00071-90066/General Utilities
00071-90065/Games

Please contact the Users' Library for further information regarding recorded material.

Solutions Book Request

Solutions Book Part Number	Title	Price
00071-900 _ _		
00071-900 _ _		
00071-900 _ _		
	Users' Library Subscription	
	State and Local Taxes	
	Solutions Book handling charge ($3.50)	
	TOTAL	

All payments must be in U.S. dollars and drawn on a U.S. bank.

Send all orders (with payment) directly to: HEWLETT-PACKARD USERS' LIBRARY
Dept. 39UL
1000 N.E. Circle Blvd.
Corvallis, Oregon 97330

☐ Payment enclosed ☐ Master Card ☐ VISA ☐ Purchase order ($20.00 minimum)

____
Card Number/Company Purchase Order

____
Name

____
Company (if applicable)

____
Address

____
City State/Country Zip

CONTENTS

Vector Operations

This program provides solutions to the most common vector operations, such as addition, subtraction, dot and cross products, included angle, multiplication of a vector by a scalar, finding the length of a vector, and determining unit vectors. The program will allow vectors to be entered in either rectangular (x, y, z) or cylindrical (r, theta, z) coordinates, and will also display the result in either form. Conversion between these two formats is also an option. A useful feature of this program is the ability to chain vector operations by using the result from one operation as an operand for the next one.

Let V1 = a1X + b1Y + c1Z and V2= a2X + b2Y + c2Z

```
         V1 + V2 = (a1+a2)X + (b1+b2)Y + (c1+c2)Z

         V1 - V2 = (a1-a2)X + (b1-b2)Y + (c1-c2)Z

         V1 X V2 = (b1*c2-b2*c1)X + (c1*a2-c1*c2)Y + (a1*b2-a2*b1)Z

         V1 . V2 = (a1*a2)X + (b1*b2)Y + (c1*c2)Z

          V1*(s) = (a1*s)X + (b1*s)Y + (c1*s)Z

        norm(V1) = (a1*a1 + b1*b1 + c1*c1)^.5

 unit vector(V1) = (a1/norm(V1))X + (b1/norm(v1))Y + (c1/norm(v1))Z

 included angle
 between V1 & V2 =
                    [ (norm(V1)^2 +norm(V2)^2-norm(V3)^2) ]
               acos [ ----------------------------------- ]
                    [        2*norm(V1)*norm(V2)          ]

                 V3 = vector between points defining V1 and V2
```

User Instructions

Comments	Input	Display
1) Run the program.		INPUT R/C ?
2) The user may enter vectors in either rectangular or cylindrical form. Press [R] to enter x, y, z components of the vector or [C] to enter r, theta and z components of the vector. If [C] is chosen, user flag 1 will be set as a visual reminder of the input format required.	[R] or [C]	DISPLAY R/C ?
3) If the result of a vector operation is a vector, the program will display the result in either rectangular or cylindrical form. Press [R] to see the x, y, and z components of the result, or press [C] to see the result expressed in terms of r, theta and z. If [C] is chosen, user flag 2 will be set as a visual reminder of the display format.	[R] or [C]	A,S,X,I,D,N,M,U,C,F,Q?
4) The program is prompting for the vector operation to be performed. The options are described below and grouped according to their input requirements. V1 represents the first vector entered, and V2 the second (if necessary).		

```
**********************************
*   SINGLE VECTOR OPERATIONS     *
**********************************
```

Comments	Input	Display
N: calculate the norm or magnitude of a vector and return the scalar result.	[N]	
M: multiply a vector by a scalar and return the vector result in the proper format.	[M]	

User Instructions

Comments	Input	Display
U: calculate the unit vector with the same direction as the input vector and with a norm of 1. Return the vector result in the proper format.	[U]	
C: convert the input vector from rectangular to cylindrical or vice versa. This option requires that the input and display formats chosen by the user correspond to the desired conversion. If they do not, the message 'I/O INCORRECT' will appear and the program will again ask for input and display formats.	[C]	
For these single vector operations, one of the following two sets of input prompts will appear, depending on the input mode selected.		
* rectangular * * coordinates *	<x>[ENDLINE] <y>[ENDLINE] <z>[ENDLINE]	x= y= z=
* cylindrical * * coordinates *	<r>[ENDLINE] <theta>[ENDLN] <z>[ENDLINE]	r= theta= z=

****
*   TWO VECTOR OPERATIONS   *
****

Comments	Input	Display
A: add V1 and V2 and return the result in the proper format.	[A]	

User Instructions

Comments	Input	Display
S: subtract V2 from V1 and return the result in the proper format.	[S]	
X: calculate the cross product V1 x V2 and return the result in the proper format.	[X]	
I: calculate the included angle between V1 and V2 and return the result in degrees.	[I]	
D: calculate the dot or scalar product V1.V2 and return the scalar result.	[D]	
For these two-vector operations, one of the following two sets of input prompts will appear, depending on the input mode selected.		
		VECTOR 1: x=
	<x1>[ENDLINE]	VECTOR 1: y=
	<y1>[ENDLINE]	VECTOR 1: z=
* rectangular *	<z1>[ENDLINE]	VECTOR 2: x=
* coordinates *	<x2>[ENDLINE]	VECTOR 2: y=
	<y2>[ENDLINE]	VECTOR 2: z=
	<z2>[ENDLINE]	
		VECTOR 1: r=
	<r1>[ENDLINE]	VECTOR 1: theta=
	<theta1>[ENDLN]	VECTOR 1: z=
* cylindrical *	<z1>[ENDLINE]	VECTOR 2: r=
* coordinates *	<r2>[ENDLINE]	VECTOR 2: theta=
	<theta2>[ENDLN]	VECTOR 2: z=
	<z2>[ENDLINE]	

```
*********************************
*        OTHER OPTIONS          *
*********************************
```

Comments	Input	Display
F: allows the user to change input and/or display formats from cylindrical to rectangular or vice versa.	[F]	<go to step 2>

Comments	Input	Display
Q: exit program.	[Q]	>
5) After an operation is chosen, the program will return the result in one of the formats listed below, depending on whether it is a vector or scalar. Pressing any key (except [ON]) will display the next component or move on to step 6.		

```
*********************************
*        VECTOR RESULTS         *
*********************************
```

[A],[S],[X],[M],[U],[C] :
These options will return a vector result in either rectangular or cylindrical coordinates, depending on the display option chosen in step 3.

Comments	Input	Display
		x=
* rectangular *	<any key>	y=
* coordinates *	<any key>	z=
	<any key>	<go to step 6>
		r=
* cylindrical *	<any key>	theta=
* coordinates *	<any key>	z=
	<any key>	<go to step 6>

```
*********************************
*        SCALAR RESULTS         *
*********************************
```

Comments	Input	Display
	[I]	INCLUDED ANGLE =
	<any key>	<go to step 6>
	[D]	DOT PRODUCT =
		<go to step 6>
	[N]	NORM =
		<go to step 6>

User Instructions

Comments	Input	Display
6) At this point, the user is given the option of exiting the program or running it again.		RUN AGAIN (Y/N)?
To exit program, press 'N'.	[N]	>
To run again, press 'Y'.	[Y]	<go to step 7>
7) If the result of the last operation was a scalar, the program will return to step 4.		
If it was a vector, the program provides the option of using it in subsequent calculations.		USE RESULT (Y/N)?
To discard the result, press 'N'. To use the result, press 'Y'.	[N] or [Y]	<go to step 4>
If 'Y' is chosen and the next operation requires one vector, the program will skip the prompt for the components and display the result. If two vectors are required, the program will assume the result is the first vector and will ask only for the second vector.		

REFERENCES:

Salas, S. and Hille, E.; "Calculus", Xerox.

Hudson, R.; "The Engineer's Manual", Wiley and Sons, Inc.

EXAMPLES

A) Find the cross product of (3,5,Ø) and (4,Ø,1), and then calculate the angle the result vector makes with the x-axis (1,Ø,Ø).

B) Find the length (norm) of the vector (12.66,-4.5,-7).

C) Convert (-4,7,Ø) to cylindrical coordinates.

Comments	Input	Display
1) Run the program.	[RUN]	INPUT R/C ?
2) Specify input in rectangular coordinates.	[R]	DISPLAY R/C ?
3) Specify output in rectangular coordinates.	[R]	A,S,X,I,D,N,M,U,C,F,Q?
4) Select cross product option.	[X]	VECTOR 1: x=
5) Enter x, y, z for vectors.	3[ENDLINE] 5[ENDLINE] Ø[ENDLINE] 4[ENDLINE] Ø[ENDLINE] 1[ENDLINE]	VECTOR 1: y= VECTOR 1: z= VECTOR 2: x= VECTOR 2: y= VECTOR 2: z=
6) Display result.	 <any key> <any key>	x=5 y=-3 z=-2Ø
7) Run program again.	<any key>	RUN AGAIN (Y/N)?
8) Use result.	[Y]	USE RESULT (Y/N)?
9) Calculate included angle.	[Y]	A,S,X,I,D,N,M,U,C,F,Q?
1Ø) Enter second vector.	[I] 1[ENDLINE] Ø[ENDLINE] Ø[ENDLINE]	VECTOR 2: x= VECTOR 2: y= VECTOR 2: z=
11) Display result.		ANGLE = 76.113ØØ63355
12) Run program again.	<any key>	RUN AGAIN? (Y/N)
13) Calculate norm.	[Y]	A,S,X,I,D,N,M,U,C,F,Q?
14) Input vector.	[N] 12.66[ENDLINE] -4.5 [ENDLINE] -7 [ENDLINE]	x= y= z=
15) Display result.		NORM = 15.15Ø1Ø23Ø99

EXAMPLES

Comments	Input	Display
16) Run program again.	<any key>	RUN AGAIN? (Y/N)
17) Convert from rectangular to cylindrical.	[Y]	A,S,X,I,D,N,M,U,C,F,Q?
18) Change output format to match conversion.	[C]	I/O INCORRECT INPUT R/C?
19) Specify rectangular input.	[R]	DISPLAY R/C?
2Ø) Specify cylindrical output.	[C]	A,S,X,I,N,M,U,C,F,Q?
21) Choose conversion now that I/O format is correct.	[C]	x=
22) Input vector.	-4 [ENDLINE] 7 [ENDLINE] Ø [ENDLINE]	y= z=
23) Display result.	 <any key> <any key>	r= 8.Ø622577483 theta= 119.744881297 z= Ø
24) End program.	<any key> [N]	RUN AGAIN? (Y/N) >

```
0010 ! VECTOR OPERATIONS
0020 ! Revision 1.0  3/28/84
0030 !
0040 DEGREES @ DELAY 0,0 @ CFLAG 1,2,3,4
0050 K0$="ASXIDNMUCFQ" @ I0$="CR" @ I1$="YN"
0060 'IO':
0070 DISP "INPUT R/C ? ";
0080 'I1': K1$=UPRC$(KEY$) @ IF NOT POS(I0$,K1$) THEN 'I1' ELSE DISP
0090 IF K1$="C" THEN SFLAG 1 ELSE CFLAG 1
0100 IF FLAG(1) THEN X1$="r=" @ Y1$="theta=" ELSE X1$="x=" @ Y1$="y="
0110 DISP "DISPLAY R/C ? ";
0120 'O1': K1$=UPRC$(KEY$) @ IF NOT POS(I0$,K1$) THEN 'O1' ELSE DISP
0130 IF K1$="C" THEN SFLAG 2 ELSE CFLAG 2
0140 IF FLAG(2) THEN X2$="r=" @ Y2$="theta=" ELSE X2$="x=" @ Y2$="y="
0150 'OP': DISP "A,S,X,I,D,N,M,U,C,F,Q"
0160 'WAIT': K1$=UPRC$(KEY$) @ K0=POS(K0$,K1$) @ IF NOT K0 THEN 'WAIT'
0170 IF K0=11 THEN 'Q'
0180 IF K0=10 THEN CFLAG 1,2,3 @ GOTO 'IO'
0190 IF K0#9 THEN 'A1'
0200 IF NOT FLAG(1) EXOR FLAG(2) THEN DISP 'I/O INCORRECT' @ WAIT 2 ELSE 'A1'
0210 GOTO 'IO'
0220 'A1': IF K0<6 THEN 'I2VA'
0230 'I1V': IF FLAG(3) THEN 'SUBCALL'
0240 DISP X1$; @ INPUT "";X1
0250 IF FLAG(1) AND X1<0 THEN DISP "INVALID ENTRY" @ GOTO 'I1V'
0260 DISP Y1$; @ INPUT "";Y1
0270 INPUT "z=";Z1
0280 GOTO 'SUBCALL'
0290 'I2VA': IF FLAG(3) THEN 'I2VB'
0300 DISP "VECTOR 1:"&X1$; @ INPUT "";X1
0310 IF FLAG(1) AND X1<0 THEN DISP "INVALID ENTRY" @ GOTO 'I2VA'
0320 DISP "VECTOR 1:"&Y1$; @ INPUT "";Y1
0330 INPUT "VECTOR 1:z=";Z1
0340 'I2VB':
0350 DISP "VECTOR 2:"&X1$; @ INPUT "";X2
0360 IF FLAG(1) AND X2<0 THEN DISP "INVALID ENTRY" @ GOTO 'I2VB'
0370 DISP "VECTOR 2:"&Y1$; @ INPUT "";Y2
0380 INPUT "VECTOR 2:z=";Z2
0390 IF FLAG(1) THEN CALL C2R(X2,Y2)
0400 'SUBCALL':
0410 IF FLAG(1) THEN CALL C2R(X1,Y1)
0420 GOSUB K1$
0430 'CYCLE':
0440 DISP "RUN AGAIN? (Y/N) ";
0450 'A2': K1$=UPRC$(KEY$) @ ON POS(I1$,K1$)+1 GOTO 'A2','USEIT','Q'
0460 'USEIT': DISP @ IF IP((K0-1)/3)=1 OR K0=9 THEN CFLAG 3 @ GOTO 'OP'
0470 DISP "USE RESULT? (Y/N)";
0480 'A3': K1$=UPRC$(KEY$) @ IF NOT POS(I1$,K1$) THEN 'A3' ELSE DISP
0490 IF POS("N",K1$) THEN CFLAG 3 @ GOTO 'OP'
0500 IF NOT POS("Y",K1$) THEN 'USEIT'
0510 IF FLAG(2) THEN CALL C2R(X0,Y0)
0520 X1=X0 @ Y1=Y0 @ Z1=Z0 @ SFLAG 3 @ GOTO 'OP'
0530 'A':
0540 X0=X1+X2 @ Y0=Y1+Y2 @ Z0=Z1+Z2
0550 GOSUB 'OV'
```

```
0560 RETURN
0570 'S':
0580 X0=X1-X2 @ Y0=Y1-Y2 @ Z0=Z1-Z2
0590 GOSUB 'OV'
0600 RETURN
0610 'D':
0620 D0=X1*X2+Y1*Y2+Z1*Z2
0630 DISP "DOT PRODUCT =";D0 @ GOSUB 'W'
0640 RETURN
0650 'X':
0660 X0=Y1*Z2-Y2*Z1 @ Y0=Z1*X2-Z2*X1 @ Z0=X1*Y2-X2*Y1
0670 GOSUB 'OV'
0680 RETURN
0690 'M': INPUT "SCALAR =";S
0700 X0=X1*S @ Y0=Y1*S @ Z0=Z1*S
0710 GOSUB 'OV'
0720 RETURN
0730 'I':
0740 SFLAG 4 @ GOSUB 'N' @ CFLAG 4
0750 M3=SQR((X2-X1)^2+(Y2-Y1)^2+(Z2-Z1)^2)
0760 A0=(M1*M1+M2*M2-M3*M3)/(2*M1*M2)
0770 C9=.00000000002
0780 IF FP(FP(ABS(A0))+C9)<.000000000045 THEN A0=IP(A0+SGN(A0)*C9)
0790 A1=ACOS(A0)
0800 DISP "INCLUDED ANGLE =";A1 @ GOSUB 'W'
0810 RETURN
0820 'N':
0830 M1=SQR(X1*X1+Y1*Y1+Z1*Z1)
0840 M2=SQR(X2*X2+Y2*Y2+Z2*Z2)
0850 IF NOT FLAG(4) THEN DISP "NORM =";M1 @ GOSUB 'W'
0860 RETURN
0870 'U':
0880 SFLAG 4 @ GOSUB 'N' @ CFLAG 4
0890 X0=X1/M1 @ Y0=Y1/M1 @ Z0=Z1/M1
0900 GOSUB 'OV'
0910 RETURN
0920 'W': IF KEY$="" THEN 'W' ELSE RETURN
0930 'C':
0940 X0=X1 @ Y0=Y1 @ Z0=Z1
0950 'OV':
0960 IF FLAG(2) THEN CALL R2C(X0,Y0)
0970 DISP X2$;X0 @ GOSUB 'W'
0980 DISP Y2$;Y0 @ GOSUB 'W'
0990 DISP "z=";Z0 @ GOSUB 'W'
1000 RETURN
1010 'Q': CFLAG 1,2,3,4 @ DISP @ PUT "#43" @ END
1020 SUB R2C(R,T)
1030 I=R @ J=T
1040 R=SQR(I*I+J*J)
1050 T=ANGLE(I,J)
1060 SUB END
1070 SUB C2R(I,J)
1080 R=I @ T=J
1090 I=R*COS(T)
1100 J=R*SIN(T)
1110 SUB END
```

Numerical Integration

This program will perform numerical integration whether a function is known explicitly or at a finite number of equally spaced points.

For the explicit case, a 16 point Gaussian quadrature is provided for a finite interval. Also, the integral may be found for the explicit case using Simpson's or the Trapezoidal methods. The quadrature is more accurate, however.

Given a finite set of points, Simpson's rule or the Trapezoidal rule may be used to find the integral.

Equations:

Gaussian Quadrature:

$$(b-a)/\ 2 \sum_{j=1}^{N} w_j f((a+b+x_j(b-a))/2)$$

where w_j, x_j are the weights and nodes, respectively. The weights and nodes are for an integral from -1 to 1.

Simpson's Rule:

$$h[f(x_0) + 4f(x_1) + 2f(x_2) + .. + 2f(x_{n-2}) + 4f(x_{n-1}) + f(x_n)]/3$$

where h is the (equally spaced) interval distance. 'n' must be even for the method to work.

Trapezoidal Rule:

$$h/2[f(x_0) + 2 \sum_{j=1}^{n=1} (f(x_j) + f(x_n))]$$

where h = (b-a)/n for interval (a,b).

User Instructions

Comments	Input	Display
1) Run the program.		Gauss,Simp,Trap,Quit
2) Choose desired method.		
a. Gauss	G	f(x)=
b. Simpson	S	Explicit (Y/N)
c. Trapezoidal	T	Explicit (Y/N)
d. Quit	Q	<blinking cursor>
Go to desired option to continue.		
****** Gauss ******		
G1) Key in the function with variable 'x'.	<function> [ENDLINE]	Lwr,Upr bounds=
G2) Key in the lower and upper bound separated by a comma.	<L,U>	calculating.. Result= <val.>
G3) To continue, press any key. Go to step 1 to continue.	<any key>	Gauss,Simp,Trap,Quit
****** Simpson ******		
S1) If the function is known: Go to the explicit case (S5).	Y	# of Partitons=<val.>
If the function is unknown:	N	Nmbr of pnts=<val.>
S2) (Implicit case) Key in the number of points. For Simpson's rule, this must be an odd number.	<number>	Interval length=<val.>
S3) Key in the interval length for the partitions. This routine requires that partitions be the same length.	<length>	E(Ø)=<val.>
You are now in the matrix editor. Continue at that section (E1) and return.		calculating.. Result=<val.>
S4) To continue, press any key. Go to step 1 to continue.	<any key>	Gauss,Simp,Trap,Quit
S5) (Explicit Case) Key in the number of partitions desired. This must be an even value.	<number>	f(x)=

User Instructions

Comments	Input	Display
S6) Key in the function with variable 'x'.	<function> [ENDLINE]	calculating.. Result= <val.>
S7) Key in the lower and upper bound separated by a comma.	<lb,ub>	calculating.. Result= <val.>
S8) To continue, press any key. Go to step 1 to continue.	<any key>	Gauss,Simp,Trap,Quit
****** Trapezoidal ******		
T1) If the function is known: Go to the explicit case (T5).	Y	# of Partitons=<val.>
If the function is unknown:	N	Nmbr of pnts=<val.>
T2) (Implicit case) Key in the number of points.	<number>	Interval length=<val.>
T3) Key in the interval length for the partitions.	<length>	E(Ø)=<val.>
You are now in the matrix editor. Continue at that section (E1) and return.		calculating.. Result=<val.>
T4) To continue, press any key. Go to step 1 to continue.	<any key>	Gauss,Simp,Trap,Quit
T5) (Explicit Case) Key in the number of partitions desired.	<number>	f(x)=
T6) Key in the function with variable 'x'.	<function> [ENDLINE]	calculating.. Result= <val.>
T7) Key in the lower and upper bound separated by a comma.	<lb,ub>	calculating.. Result= <val.>
T8) To continue, press any key. Go to step 1 to continue.	<any key>	Gauss,Simp,Trap,Quit

Comments	Input	Display
****** Matrix Editor ******		
The points E(Ø) through E(n) may be entered. The value displayed is the current value of the point. You may change the value, move to a previous value, move to the next value, or specify a point to move to. Pressing [Q] exits the matrix editor, and the program finds the value of the integral.		E(I)=<value>
E1) To change value:	[ENDLINE] <new val.> [ENDLINE]	 E(I+1)=<value>
To move to next value (this is the arrow key, not the greater than char.):	[>]	E(I+1)=<value>
To move to previous value (this is the arrow key, not the less than char.):	[<]	E(I-1)=<value>
To move to a specific point:	[SPC]	Element=
Key in the desired element index:	<index>	E(<index>)=<val.>
To quit the routine:	[Q]	calculating..

EXAMPLE PROBLEMS

Problem 1.

Given the approximations below for f(x), compute the approximations to the integral from the bounds Ø-2 by the trapezoidal rule and by Simpson's rule. The interval length is Ø.25.

i	Ø	1	2	3	4	5	6	7	8
f(x)	2	2.8	3.8	5.2	7	9.2	12.1	15.6	2Ø

Comments	Input	Display
1) Run the program.		Gauss,Simp,Trap,Quit
2) Choose trapezoidal method:	T	Explicit (Y/N)
3) Since the function is unknown, choose implicit case:	N	Nmbr of pnts=Ø
4) Key in the number of points.	9[ENDLINE]	Interval length=Ø
5) Key in the interval length for the partitions.	.25[ENDLINE]	E(Ø)=Ø
6) Enter the function values:	[ENDLINE]	value=
	2	
	[ENDLINE]	E(1)=Ø
	[ENDLINE]	value=
	2.9	
	[ENDLINE]	E(2)=Ø
The previous value was supposed to have been 2.8. Edit it.	[<]	E(1)=2.9
	[ENDLINE]	value =
	2.8	
	[ENDLINE]	E(2)=Ø
Continue entering the rest of of the values in the same manner.		
To quit:	Q	calculating.. Result=16.675Ø
7) To continue, press any key.	<any key>	Gauss,Simp,Trap,Quit
8) Now use Simpson's rule.	S	Explicit (Y/N)
9) Since the function is unknown, choose implicit case:	N	Nmbr of pnts=9

User Instructions

Comments	Input	Display
1Ø) Since this is the correct number, just press:	[ENDLINE]	Interval length=.25ØØ
11) This value is also okay.	[ENDLINE]	E(Ø)=2
12) Since the program leaves the values the same, simply press [Q] to finish.	Q	calculating.. Result=16.5833
13) To continue, press any key.	<any key>	Gauss,Simp,Trap,Quit

Problem 2.

Find the value of the integral of f(x) from Ø to 2*pi where
f(x) = 1/(1-cos(x)+ .25).

Comments	Input	Display
1) Run the program.		Gauss,Simp,Trap,Quit
2) Choose Gauss's method:	G	f(x)=
3) Key in the function with variable 'x'.	1/(1-cos(x)+.25) [ENDLINE]	Lwr,Upr bounds=
4) Key in the lower and upper bound separated by a comma.	Ø,2*pi [ENDLINE]	calculating.. result= 8.3776
5) To continue, press any key.	<any key>	Gauss,Simp,Trap,Quit

LISTING

Note that some of the comments are not preceded by line numbers.

```
0010 DEF FNK$(D0$,K0$)     ! Find which key was pressed function
0020 DISP D0$
0030 'KEY': K$=KEY$
0040 IF NOT POS(K0$,K$) THEN "KEY"
0050 FNK$=K$
0060 END DEF

**************   INITIALIZATION *************

0070 OPTION BASE 0
0080 DESTROY A,H,P,T$,K$,K1$
0090 DIM K$[4],K1$[1],T$[1]
0100 INTEGER N,P

**** BEGIN ****

0110 'START': K1$=FNK$("Gauss,Simp,Trap,Quit","GSTQ")
0120 GOTO K1$

**** SIMPSON, TRAPEZOIDAL START ****

0130 'S': 'T': T$=FNK$("Explicit?(Y/N)","YN")
0140 IF T$="Y" THEN "PRT"

**** For implicit case, get function values

0150 'PL': INPUT "Nmbr of pnts= ",STR$(N);N
0160 IF N<=0 THEN DISP "Must be positive" @ GOTO 'PL'
0170 IF K1$="S" AND MOD(N-1,2) THEN DISP "MUST BE ODD NUMBER" @ GOTO
"PL"
0180 INPUT "Interval Length= ",STR$(H);H
0190 DIM A(N-1)
0200 CALL EDIT1(A,N-1)
0210 GOTO "CALC"

**** For explicit case, get number of partitions ****

0220 'PRT': INPUT "# of Partitions= ",STR$(P);P
0230 IF P<=0 THEN DISP "Must be positive" @ GOTO 'PRT'
0240 IF K1$="S" AND MOD(P,2) THEN DISP "MUST BE EVEN, NONZERO" @ GOTO
"PRT"

**** For explicit case, get function ****

0250 'G': INPUT "f(x)= ",F$;F$
0260 IF K1$="G" THEN T$="N"
0270 IF F$="" THEN "G"

**** Get the bounds for the integral ****

0280 'B': INPUT "Lwr,Upr bounds= ",STR$(L)&","&STR$(U);L,U
0290 IF L=U THEN DISP "THE INTEGRAL VALUE IS 0" @ GOTO "B"
```

LISTING

```
0300 'CALC': O=FLAG(-16,O) @ T=FLAG(-10,1) ! Temporarily set radians
0310 DISP "calculating.."
0320 GOSUB K1$&T$
0330 O=FLAG(-16,O) @ T=FLAG(-10,T) ! Re-establish angular mode
0340 D1$=PEEK$("2F946",4) @ DELAY INF,INF ! Save wait period
0350 DISP "Result=";R ! Display result
0360 POKE "2F946",D1$ ! Re-establish wait period
0370 GOTO "START"
```

****** SUB PROGRAM CALL SECTION ******

```
0380 'GN': CALL GAUSS(F$,L,U,R) @ RETURN
0390 'SN': CALL SIMPSON(A,N-1,H,R) @ RETURN
0400 'TN': CALL TRAP(A,N-1,H,R) @ RETURN
0410 'SY': CALL SIMPSONE(F$,L,U,(P),R) @ RETURN
0420 'TY': CALL TRAPE(F$,L,U,(P),R) @ RETURN
0430 'Q': PUT "#38" @ END  ! Restore cursor and end program
0440 !
```

***** Trapezoidal rule, explicit case ****

```
0450 SUB TRAPE(F$,L,U,P,R)
0460 H=(U-L)/P
0470 Q=0
0480 X=L @ R=VAL(F$)
0490 X=U @ R=R+VAL(F$)
0500 FOR I=1 TO P-1
0510 X=L+H*I ! what is A?
0520 Q=Q+VAL(F$)
0530 NEXT I
0540 R=(R+2*Q)*H/2
0550 END SUB
0560 !
```

**** Trapezoidal rule, implicit case ****

```
0570 SUB TRAP(A(),N,H,R)
0580 R=A(0)+A(N)
0590 Q=0
0600 FOR I=1 TO N-1
0610 Q=Q+A(I)
0620 NEXT I
0630 R=(R+2*Q)*H/2
0640 END SUB
0650 !
```

**** Simpson's rule, implicit case ****

```
0660 SUB SIMPSON(A(),N,H,R)
0670 R=A(0)+A(N)
0680 FOR I=1 TO N-3 STEP 2
0690 R=R+4*A(I)+2*A(I+1)
0700 NEXT I
0710 R=(R+4*A(N-1))*H/3
```

```
0720 END SUB
0730 !
```

**** Simpson's rule, explicit case ****

```
0740 SUB SIMPSONE(F$,L,U,P,R)
0750 H=(U-L)/P
0760 X=L @ R=VAL(F$)
0770 X=U @ R=R+VAL(F$)
0780 FOR I=1 TO P-3 STEP 2
0790 X=L+H*I @ R=R+4*VAL(F$)
0800 X=X+H @ R=R+2*VAL(F$)
0810 NEXT I
0820 X=U-H
0830 R=(R+4*VAL(F$))*H/3
0840 END SUB
0850 !
```

**** 16 point Gaussian method ****

```
0860 SUB GAUSS(F$,L,U,R)
0870 DIM N(1,7)
0880 DATA .989400934992,.944575023073,.865631202388,.755404408355
0890 DATA .617876244403,.458016777657,.281603550779,.950125098376E-1
0900 DATA .271524594118E-1,.622535239386E-1,.951585116825E-1,.124628971256
0910 DATA .149595988817,.169156519395,.182603415045,.189450610455
0920 READ N(,)
0930 R=0
0940 C=(U-L)/2
0950 FOR I=0 TO 7
0960 X=N(0,I)*C+(L+U)/2
0970 R=N(1,I)*VAL(F$)+R
0980 X=-N(0,I)*C+(L+U)/2
0990 R=N(1,I)*VAL(F$)+R
1000 NEXT I
1010 R=R*C
1020 END SUB
```

**** Matrix Editor ****

```
1030 SUB EDIT1(A(),N)
1040 DEF FNK$(D0$,K0$)   ! Find which key was pressed function
1050 DISP D0$
1060 'KEY': K$=KEY$
1070 IF NOT POS(K0$,K$) THEN "KEY"
1080 FNK$=K$
1090 END DEF

1100 DEF FNF$(Y)     ! Fix display function. Temporarily alters,
                     ! then returns to original setting.
1110 D$=PEEK$("2F6DC",2) @ STD
1120 FNF$=STR$(Y)
1130 POKE "2F6DC",D$
1140 END DEF
```

LISTING

```
1150 I=0
1160 'LOOP': K1$=FNK$("E("&FNF$(I)&")= "&STR$(A(I)),"Q #38#47#48")
1170 IF K1$=" " THEN 'M'
1180 IF K1$="Q" THEN 'Q'
1190 IF LEN(K1$)#3 THEN 'LOOP'
1200 K1$[1,1]="K"
1210 GOSUB K1$
1220 GOTO 'LOOP'
1230 'K47': I=MOD(I-1,N+1)        ! Move to previous position
1240 RETURN
1250 'K38': INPUT "Value= ";A(I)  ! Change value
1260 'K48': I=MOD(I+1,N+1)        ! Move forward
1270 RETURN

1280 'M': INPUT "Element= ";I     ! Get desired position
1290 IF I<0 OR I>N THEN DISP "EXCEEDS BOUNDS" @ WAIT 1 @ GOTO 'M'
1300 GOTO "LOOP"

1310 'Q': PUT "#38"               ! Quit routine
1320 END SUB
```

Solution to F(x) = 0 on an Interval

This program provides two methods to find a real root of the equation f(x) = 0. They are Newton's Method and the Pegasus Method. In addition, the program allows the user to find the value of the function for an input x.

Input for the Newton's method consists of the function to be solved, one initial guess, and as an option, the derivative of the function. If the derivative is not input, a numerical approximation is used.

Input for the Pegasus method consists of the function to be solved and two initial guesses that must bound the root. This implies that f(x0)*f(x1)<0. The routine to calculate function values may be used to establish a legal interval.

When a root is found, the output will consist of the x value displayed to the setting of the computer and the value f(x), where x is the displayed root. It is possible that f(x) will not be exactly 0. However, it will generally be within an acceptable range around zero based on the number of significant digits in the input. If the desired accuracy is not obtained, it may be possible to decrease the value used to check for acceptance (the variable E in both sub programs). In some instances, the function may have to be modified.

Newton's method converges to a root quickly in cases where it can find one. Its ability to locate a root depends on the function and the initial guess. It is not guaranteed to find a root. If the derivative is 0 or 50 iterations are peformed, the routine exits, displaying an appropriate message to the user. The number of iterations may be changed by altering the program.

The Pegasus Method is a modified regula falsi method with an estimated order of convergence superior to a secant method. For any legal interval, the method is guaranteed to converge.

The equations:

Newton's Method:

x<n+1> = x<n>-f(x<n>)/f'(x<n>)

The exit criteria is abs(x[n+1]-x[n]) <= epsilon where epsilon is a small value.

f'(x) Approximation:

When the derivative is not given, the program uses the following approximation:

f'(x) <== (f(x+I/2) - f(x-I/2))/I

where I = .0001(x) or .000001 if x= 0.

Pegasus Method:

The Regula Falsi method used is:

```
x<n+1> = x<n> - f(x<n>)[(x<n>-x<n-1>)/(f(x<n>)-f(x<n-1>))]
```

The approximations for the next iteration are chosen by:

```
if f(x<n+1>)f(x<n>) < 0 then
 (x<n-1>,f(x<n-1>)) <== (x<n>, f(x<n>))
if f(x<n+1>)f(x<n>) > 0 then
 (x<n-1>,f(x<n-1>)) <== (x<n-1>, f(x<n-1>)/(f(x<n>)+f(x<n+1>)))
```

References:

Dowell, M. & Jarrett, P.; "The Pegasus Method for Computing the Root of an Equation", BIT 12 (1972) pp. 503-508

Atkinson, Kendall E., "An Introduction to Numerical Analysis", Wiley and Sons

Carnahan, B., Luther, H.A., and Wilkes, J.O., "Applied Numerical Methods", Wiley and Sons, Inc.

USER INSTRUCTIONS

Comments	Input	Display
1) Run program.		f(x)=
2) Key in the function using the character 'x' as the variable.	<function>	Root,F(x),Chngf,Quit
3) Press the capital letter of the desired operation.		
'R' begins the solve routine.	R	Pegasus, Newton
'F' finds function values for given x's.	F	X=
'C' allows the function to be changed.	C	f(x)=<function>
'Q' quits the program.	Q	Done
Go to the appropriate heading to continue.		
****** ROOT OPTION ******		Pegasus, Newton
R1) Choose desired method by pressing the appropriate capital letter.		
For Pegasus, press P. Go to step R2.	P	lower bound:
For Newton, press N. Go to step R8.	N	derivative=
R2) PEGASUS METHOD		lower bound:
R3a) Key in a lower bound.	<value> [ENDLINE]	upper bound:
R3b) To exit, press:	[ENDLINE]	Root,F(x),Chngf,Quit
R4a) Key in upper bound.	<value> [ENDLINE]	calculating..
R4b) To exit, press:	[ENDLINE]	Root,F(x),Chngf,Quit
R5a) The answer will be displayed.		x= <result>
R5b) If the interval does not bound a root a message will be displayed and you will return to step R4.		
R6) To see the function value at the root, press any key.	<any key>	f(x)=<value>

USER INSTRUCTIONS

Comments	Input	Display
R7) To exit, press any key.	<any key>	Root,F(x),Chngf,Quit
R8) NEWTON METHOD		derivative =
R9a) If you want to use the derivative, key it in.	<derivative> [ENDLINE]	initial guess=
R9b) If you do not wish to enter the derivative:	[ENDLINE]	initial guess=
R1Ø) Key in initial guess.	<guess> [ENDLINE]	disp convergence?
R11) If you wish to watch the convergence, press 'Y', else press any other key.	Y or <another key>	calculating..
R12) If a root is found, the result will be found. If a root is not found, an appropriate error message will be given.		x= <result>
R13) To see the function value at the root, press any key.	<any key>	f(x)=<value>
R14) To exit, press any key.	<any key>	Root,F(x),Chngf,Quit
******* FIND FUNCTION VALUES ******		X=
F1) Key in the value of x at which you wish the function evaluated at.	<value> [ENDLINE]	f(x) = <result>
F2) Press any key to continue.	<any key>	X=
F3a) Continue at step F2 for more values.		
F3b) To exit, press:	[ENDLINE]	Root,F(x),Chngf,Quit
****** CHANGE FUNCTION ******		f(x)=<crnt function>
C1) Key in desired function.	<function> [ENDLINE]	Root,F(x),Chngf,Quit
****** QUIT PROGRAM ******		Done
Q1) To get the prompt back: press any key.	<any key>	<blinking prompt>

EXAMPLE

This example demonstrates the various options of the program. It must be started at the beginning and followed to completion for the given keystrokes to work as listed. The display should be FIX 11.

Find a root for the function f(x) = ln(x) + 3x - 1Ø.8Ø74. Use the Pegasus method, then Newton's method. Then find a root for the function f(x) = 3x^6 - 22x^5 + 11.

Comments	Input	Display
1) Run program.		f(x) =
2) Key in function. Note that it is entered in a BASIC format.	ln(x)+3*x-1Ø.8Ø74 [ENDLINE]	 Root,F(x),Chngf,Quit
3) Call the solve section.	R	Pegasus,Newton
4) Choose Pegasus method.	P	lower bound:
5) Guess at a bound:	5 [ENDLINE]	upper bound:
6) Guess at an upper bound.	1Ø [ENDLINE]	intrvl must bound root lower bound:
7) The interval did not bound the root. Use F(x) to find an interval.	[ENDLINE] F	Root,F(x),Chngf,Quit x=
8) Find f(5).	5 [ENDLINE] [ENDLINE]	f(x) = 5.8Ø2Ø379124 x=
9) Find f(1Ø)	1Ø [ENDLINE] [ENDLINE]	21.495185Ø93 x=
1Ø) Try f(1)	1 [ENDLINE] [ENDLINE]	f(x)= -7.8Ø74 x=
11) Since the function is continuous on the interval [1,5], and the values of the function at these points are of opposite sign, this interval bounds a root. Exit and move to the solve section.	[ENDLINE] R P	Root,F(x),Chngf,Quit Pegasus,Newton lower bound:
12) Key in lower bound	1 [ENDLINE]	upper bound:

EXAMPLE

Comments	Input	Display
13) Key in upper bound.	5 [ENDLINE]	calculating.. x= 3.21336Ø87Ø18
14) See how close f(3.21..) is to Ø.	[ENDLINE]	f(x) = Ø
15) Continue	[ENDLINE]	Root,F(x),Chgf,Quit
16) Solve using Newton's Method.	R N	Pegasus, Newton derivative=
17) Let routine approximate the derivative.	[ENDLINE]	initial guess=
18) Key in an initial guess.	1 [ENDLINE]	disp convergence?
19) Let's not display it.	N [ENDLINE] [ENDLINE]	calculating.. x= 3.21336Ø87Ø17 f(x)= Ø Root,F(x),Chngf,Quit
2Ø) We must now solve the second function.	C 3*x^6-22*x^5+11 R N	f(x)=LN(X)+3*X-1Ø.8Ø74 Root,F(x),Chngf,Quit Pegasus,Newton derivative=
21) Use the derivative this time. The intermediate results are displayed.	18*x^5-11Ø*x^4 [ENDLINE] 5 [ENDLINE] Y [ENDLINE] [ENDLINE]	 initial guess= disp convergence? calculating.. 2.25Ø88 2.4794542177 1.93158885Ø68 . . .89346784412 x= .893467844Ø31 f(x)= Ø Root,F(x),Chngf,Quit
22) Quit the program.	Q	DONE

LISTING

Note that some of the comments are not preceded by a line number.

```
0010 ! SOLUTION TO F(X)=0
0020 ! REV 1.0 -- 1/25/84
0030 F$=""
0040 'C': INPUT "f(x)=",F$;F$ ! get function
0050 'D': DISP "Root,F(x),Chngf,Quit" ! main prompt
0060 A$=KEY$ @ IF A$="" THEN 60
0070 IF POS("RFQC",UPRC$(A$[1,1]))THEN GOTO UPRC$(A$[1,1]) ELSE GOTO 'D'

******* Find the Root *******

0080 'R': DISP "Pegasus,Newton" ! get desired method
0090 M$=KEY$ @ IF M$="" THEN 90
0100 IF POS("PN",UPRC$(M$[1,1])) THEN GOSUB M$[1,1] ELSE GOTO 'R'
0110 IF R<>INF THEN GOTO 'DISPR'
0120 DISP "NO ROOT FOUND"
0130 GOTO 'RTN'
0140 'DISPR': X=R ! display result
0150 DISP "x= ";X
0160 A$=KEY$ @ IF A$="" THEN 160
0170 DISP "f(x)= ";VAL(F$) ! display function value
0180 'RTN': A$=KEY$ @ IF A$="" THEN GOTO 'RTN' ELSE GOTO 'D'

******* Find values of f(x) given x *******

0190 'F': ON ERROR GOTO 'XIT'
0200 INPUT "X= ";X
0210 Y=VAL(F$)
0220 DISP "f(x)= ";Y
0230 A$=KEY$ @ IF A$="" THEN 230 ELSE "F"
0240 GOTO 'F'
0250 'XIT': OFF ERROR @ GOTO 'D'

******* Set up to call Pegasus method *******

0260 'P': ON ERROR GOTO 'XIT'
0270 INPUT "lower bound:";L
0280 INPUT "upper bound: ";U
0290 X=L ! see if interval contains root
0300 Y1=VAL(F$)
0310 X=U
0320 Y2=VAL(F$)
0330 IF Y1*Y2>0 THEN DISP "intrvl must bound root" @ GOTO 'P'
0340 CALL PEG(F$,L,U,R)
0350 RETURN

******* Set up to call Newton method *******

0360 'N': DESTROY N$
0370 ON ERROR GOSUB 'NEWTERR' ! catches no derivative option
0380 INPUT "derivative= ";D$
0390 INPUT "initial guess= ","1";X0
0400 DISP "disp convergence?"
```

```
0410 A$=KEY$ @ IF A$="" THEN 410
0420 IF UPRC$(A$)="Y" THEN D=1 ELSE D=0
0430 CALL NEWT(F$,D$,X0,D,R)
0440 RETURN
0450 'NEWTERR': OFF ERROR
0460 ON ERROR GOTO 'XIT'
```

******* Quit program *******

```
0470 'Q': DISP " DONE" @ END
```

******* Pegasus subprogram *******
The inputs are the function (F$), lower limit (X0), and upper limit (X1). The result is returned through variable 'R'. The function string must be a legal BASIC expression in variable 'x'; the limits must bound a root.

```
0480 SUB PEG(F$,X0,X1,R)
0490 DISP "calculating.."
0500 E=.0000000001 ! error tolerance
0510 C=2*X1-X0
0520 X=X0 ! init y0,y1
0530 Y0=VAL(F$)
0540 X=X1
0550 Y1=VAL(F$)
0560 !
0570 'LP': X2=X1-Y1*((X1-X0)/(Y1-Y0))
0580 IF ABS(X2-C)<=E THEN R=X2 ELSE GOTO 'CT'
0590 GOTO 'ND'
0600 'CT': X=X2
0610 Y2=VAL(F$)
0620 C1=Y2*Y1
0630 IF C1<0 THEN X0=X1 @ Y0=Y1
0640 IF C1>0 THEN Y0=Y0*Y1/(Y1+Y2)
0650 X1=X2
0660 Y1=Y2
0670 C=X2
0680 GOTO 'LP'
0690 'ND': SUB END
```

******* Newton subprogram *******
The inputs are the function (F$), optional derivative (D$), initial guess, and the display-convergence boolean. The result is returned through 'R'. The functions must be in variable 'x'. If no root is found, 'R' is set to the value 'inf'.

```
0700 SUB NEWT(F$,D$,X0,D,R)
0710 DISP "calculating.."
0720 E=.0000000001 ! error tolerance
0730 L=0 ! INIT LOOP COUNTER
0740 'LP': IF D$="" THEN GOSUB 'AD' ELSE X=X0 @ Y1=VAL(D$)
0750 IF Y1=0 THEN R=INF @ DISP "DERIVATIVE=0" @ GOTO 'ND'
0760 X=X0
0770 Y=VAL(F$)
```

```
0780 X1=X-Y/Y1
0790 IF ABS(X1-X)<=E THEN R=X1 @ GOTO 'ND'
0800 X0=X1
0810 L=L+1
0820 IF D THEN DISP X0
0830 IF L=50 THEN DISP "50 ITERATIONS" @ R=INF @ GOTO 'ND'
0840 GOTO 'LP'
0850 'AD': IF X0=0 THEN I=.000001 ELSE I=.0001*X0 ! deriv. approx.
0860 X=X0+I/2
0870 Y1=VAL(F$)
0880 X=X0-I/2
0890 Y1=(Y1-VAL(F$))/I
0900 RETURN
0910 'ND': SUB END
```

This program allows the user to caculate the determinant of a real valued matrix and find the inverse or solve a system of equations for real or complex valued systems.

The method used is Gaussian elimination with partial pivoting. The matrix is decomposed into an LU form and the pivoting strategy is kept track of in a separate matrix. For a more in-depth discussion, please refer to the references.

The determinant is calculated from the decomposed matrix by multiplying the values in the main diagonal. It is found only for a real valued matrix.

The inverse is found by solving the system Ax(i) = I(i), where I(i) is the ith column of the identity matrix and x(i) is the ith column of A inverse. This is performed N times as i ranges from 1 to N.

If the inverse or solution to a system of equations is attempted that involves a singular matrix, the message "MATRIX IS SINGULAR" will be displayed.

Remarks:

The program will use the particular number display you specify before running the program.

If you pause the program during the listing of values, then exit the program, your delay will be set to inf.

References:

Johnston, R.L., "NUMERICAL METHODS, a Software Approach", John Wiley and Sons, 1982

Anton, Howard, "Elementary Linear Algebra", John Wiley and Sons, 1981

Atkinson, Kendal E., "An Introduction to Numerical Analysis", John Wiley and Sons, 1978

USER INSTRUCTIONS

Comments	Input	Display
1) Run the program.		Real or Complex?
2) For real values:	R	
For complex values:	C	order?
3) Key in the number of rows: The matrix must be square.	<value>	A(1,1)=<value> or R(1,1)=<value>
You are now in the matrix editor. Please refer to that section, then continue with step 4.		keep a copy?
4) If you want to perform some operations and then make a few changes to the matrix, keep a copy of the original:	Y	
If no copy is desired:	N	Newmat,Edit,Calc,Quit
5) Choose desired action.		
To create a new matrix: Cont. at step 1.	N	Real or Complex?
To edit the current matrix: Refer to the editor instructions, then cont. at step 4.	E	A(1,1)=<value> or R(1,1)=<value>
To perform matrix operations: Continue at step 6.	C	Solv,Det,Inv,Main,Quit
To quit the program:	Q	<blinking prompt>
6) Calculation options.		
To solve system of equations:	S	B(1)=<value>
You are now in the matrix editor. Enter the values of the B vector for Ax=B. When you are done, you will see: Continue at solve section.		calculating..
To find the determinant: Continue at the determinant section.	D	calculating..
To find the inverse: Cont. at the inverse secton.	I	calculating..
To return to the main menu: Continue at step 5.	M	Newmat,Edit,Calc,Quit
To quit the program:	Q	<blinking display>

USER INSTRUCTIONS

Comments	Input	Display
****** SOLVE ******		
S1) If a solution exists:		<value x(1)>
These are the values of	[ENDLINE]	<value x(2)>
the result vector.	[ENDLINE]	.
	.	.
Continue at step S2.		
If a solution does not exist, a message will be displayed:		SINGULAR SYSTEM Newmat,Edit,Calc,Quit
Return to step 6.		
S2) Once the last value has been displayed, you will see:		solve for new B?
S3) If you want to solve for a new B vector:	Y	B(1)=<value>
Enter the values according to the edit instructions.		
Continue at step S1.		
If you are done:	N	Solv,Det,Inv,Main,Quit
Return to step 6.		
****** DETERMINANT ******		
D1) The determinant will be dislayed:		det = <value>
Press any key to continue.	<any key>	Solv,Det,Inv,Main,Quit
Go to step 6.		
****** INVERSE ******		
I1) If the inverse exists:		list by Col or Row?
Cont. at step I2.		
If no inverse exists:		SINGULAR SYSTEM Newmat,Edit,Calc,Quit
Return to step 6.		
I2) To list result by columns:	C	<value A(1,1)>
	[ENDLINE]	<value A(1,2)>
	.	.
	.	.
To list result by rows:	R	<value A(1,1)>
	[ENDLINE]	<value A(1,2)>
	.	.
	.	.
I3) Once all values have been listed, you will see:		Solv,Det,Inv,Main,Quit
Go to step 6.		

****** MATRIX EDITOR ******

The matrix editor allows the user to move through a matrix and change element values. For complex valued matrices, the real and complex parts of an element are edited one at a time.

Movement through a matrix is accomplished with the arrow keys (left, right, up, and down), and element indices input. The movement wraps around when the boundary of a row or column is passed.

Comments	Input	Display
E1) To move from the current position:		A(I,J)
Move left:	<	A(I,J-1)= <value>
Move right:	>	A(I,J+1)= <value>
Move up:	<up arrow key>	A(I-1,J)= <value>
Move down:	<dwn arrow key>	A(I+1,J)= <value>
E2) To move to a desired element:	[SPC]	enter ROW,COLUMN
	<row>,<column>	
	[ENDLINE]	A(<row>,<col>)= <val>
E3) To quit the editor:	Q	

EXAMPLES

Example 1.

Find the determinant and inverse of the matrix below. It is assumed that the display is fix 4.

6	3	-2	2	3
1	4	-3	4	2
2	3	-1	-2	9
4	3	Ø	2	1
3	5	-6	6	2

Comments	Input	Display
1) Run the program.		Real or Complex?
2) The values are real.	R	order?
3) Key in the number of rows:	5[ENDLINE]	A(1,1)= Ø.ØØØØ
	[ENDLINE]	?
	6	
	[ENDLINE]	A(1,2)= Ø.ØØØØ
	[ENDLINE]	?
	3	
Enter the values of the matrix and use the editing features to adjust any incorrect values.	[ENDLINE]	A(1,3)= Ø.ØØØØ
	[ENDLINE]	?
	-2	
	[ENDLINE]	A(1,4)= Ø.ØØØØ
	[ENDLINE]	?
	2	
	[ENDLINE]	A(1,5)= Ø.ØØØØ
	[ENDLINE]	?
	3	
	.	.
	.	.
	[ENDLINE]	A(5,4)= Ø.ØØØØ
	[ENDLINE]	?
	6	
	[ENDLINE]	A(5,5)= Ø.ØØØØ
	[ENDLINE]	?
	2	
	[ENDLINE]	A(1,1)= 6.ØØØØ
	Q	keep a copy?
4) No copy will be needed.	N	Newmat,Edit,Calc,Quit
5) Choose calculation options.	C	Solv,Det,Inv,Man,Quit

6) Calculate the determinant.	D	calculating.. det = -2ØØ.ØØØØ
Press any key to continue.	[ENDLINE]	Solv,Det,Inv,Main,Quit
7) To find the inverse:	I	calculating.. list by Col. or Row?
8) Choose to list by column.	C	C(1,1)= Ø.2ØØØ
	[ENDLINE]	C(2,1)=-1.75ØØ
	[ENDLINE]	C(3,1)= Ø.7ØØØ
	[ENDLINE]	C(4,1)= 1.725Ø
	[ENDLINE]	C(5,1)= 1.ØØØØ
	[ENDLINE]	C(1,2)=-Ø.12ØØ
	[ENDLINE]	C(2,2)=-2.ØØØØ
	[ENDLINE]	C(3,2)= 1.28ØØ
	[ENDLINE]	C(4,2)= 2.54ØØ
	[ENDLINE]	C(5,2)= 1.4ØØØ
	[ENDLINE]	C(1,3)=-Ø.Ø4ØØ
	[ENDLINE]	C(2,3)= Ø.5ØØØ
	[ENDLINE]	C(3,3)=-Ø.24ØØ
	[ENDLINE]	C(4,3)=-Ø.57ØØ
	[ENDLINE]	C(5,3)=-Ø.2ØØØ
	[ENDLINE]	C(1,4)= Ø.ØØØØ
	[ENDLINE]	C(2,4)= 1.75ØØ
	[ENDLINE]	C(3,4)=-Ø.5ØØØ
	[ENDLINE]	C(4,4)=-1.625Ø
	[ENDLINE]	C(5,4)=-1.ØØØØ
	[ENDLINE]	C(1,5)=-3.3333E-12
	[ENDLINE]	C(2,5)=-1.5ØØØ
	[ENDLINE]	C(3,5)=-1.ØØØØ
	[ENDLINE]	C(4,5)=-1.75ØØ
	[ENDLINE]	C(5,5)=-1.ØØØØ
	[ENDLINE]	Solve,Det,Inv,Main,Quit
9) Now find the determinant.	D	calculating.. det = -2ØØ.ØØØØ
	[ENDLINE]	Solve,Det,Inv,Main,Quit
1Ø) Return to the main menu to perform the next problem.	M	Newmat,Edit,Calc,Quit

Problem 2.

Solve the following system of complex valued equations.

$$\begin{matrix} 2 + 3i & .7 - 1i \\ 4 - 1.3i & 4 + 0i \end{matrix} \quad \begin{matrix} z1 \\ z2 \end{matrix} \quad = \quad \begin{matrix} 2 + 21i \\ 1 + 3i \end{matrix}$$

Because the matrix created in problem 1 is real valued, a new matrix must be created. Therefore the example problem must start with the new matrix option.

EXAMPLES

Comments	Input	Display
1) Start by creating a new		
matrix.	N	Real or Complex?
2) Choose complex.	C	order?
3) The order is 2.	2	
	[ENDLINE]	R(1,1)= 0.0000
4) You are now in the		
editor.		
The 'R' indicates that	[ENDLINE]	?
the real portion of	2	
element (1,1) is being	[ENDLINE]	I(1,1)= -6.0000
prompted for.	[ENDLINE]	?
'I' indicates that the	3	
imaginary portion is	[ENDLINE]	R(1,2)= 3.0000
being prompted for.	[ENDLINE]	?
	.7	
	[ENDLINE]	I(1,2)= 2.0000
Continue entering the	.	.
values in the same manner.	.	.
Refer to the editor	.	.
instructions for editing	.	.
features.	.	.
5) When done, press:	Q	keep a copy?
6) Keep a copy this time.	Y	Newmat,Edit,Calc,Quit
7) Move to calculation options.	C	Solv,Det,Inv,Main,Quit
8) Choose solve option.	S	BR(1)=4.0000
9) You are now in the editor.The		
prompt is for the real portion		
of element (1,1) from the		
b vector (Ax = b).	[ENDLINE]	?
	2	
	[ENDLINE]	BI(1)= 0.0000
	[ENDLINE]	?
	21	
	[ENDLINE]	BR(2)= 0.0000
Continue until all	.	.
values are correctly entered.	.	.
	.	.
10) To quit editing:	Q	calculating..
		X(1)= 4.3565, 1.4203 i
You may use the arrow	[ENDLINE]	X(2)=-4.5681, 0.7456 i
to view the entire display.	[ENDLINE]	solve for new B?

EXAMPLES

Comments	Input	Display
11) Let's solve for a new b		
vector.	Y	BR(1)= 1.0000
	[ENDLINE]	?
	9	
	[ENDLINE]	BI(1)= 21.0000
	[ENDLINE]	?
	-.22	
	[ENDLINE]	BR(2)= 1.4794
	[ENDLINE]	?
	-3.5	
	[ENDLINE]	BI(2)= 3.0000
	1	
	[ENDLINE]	BR(1)= 9.0000
	Q	calculating..
		X(1)= 0.4800,-2.0302 i
	[ENDLINE]	X(2)=-0.6952, 2.4362 i
	[ENDLINE]	solve for new B?
12) Let's exit.	N	Solv,Det,Inv,Main,Quit
	Q	<blinking display>

PROGRAM LISTING

```
0010 ! rev 1.0
0020 DEF FNK$(D0$,K0$)   ! Find which key was hit
0030 DISP D0$
0040 'KEY': K$=KEY$
0050 IF NOT POS(K0$,K$) THEN "KEY"
0060 FNK$=K$
0070 END DEF

0080 'N': T=POS("RC",FNK$("Real or Complex?","RC"))
0090 INPUT "order?";N ! Get the number of rows and columns.

0100 N=T*N @ M=N ! Set dimensions based on type.
0110 DIM A(M,N),S(M)
0120 R2=0 @ C2=0

0130 'E': IF NOT C2 THEN GOTO 'E1'
0140 FOR I=1 TO M
0150 FOR J=1 TO N
0160 A(I,J)=A1(I,J)   ! Read in original matrix if copy was made
0170 NEXT J
0180 NEXT I

0190 'E1': CALL EDIT(A(,),M,N,T)
0200 R2=0     ! Indicate that the determinant has not been calculated.
0210 IF FNK$("keep a copy?","YN")="Y" THEN C2=1 ELSE C2=0
0220 IF NOT C2 THEN GOTO 'M'
0230 DIM A1(M,N)
0240 FOR I=1 TO M
0250 FOR J=1 TO N
0260 A1(I,J)=A(I,J) ! Save a copy if desired.
0270 NEXT J
0280 NEXT I

0290 'M': GOTO FNK$("Newmat,Edit,Calc,Quit","NECQ") ! main menu

0300 'C': GOTO FNK$("Solv,Det,Inv,Main,Quit","SDIMQ")

0310 'S': DIM B(M,T),X(M,1) ! solve invocation.
0320 'S1': CALL EDIT(B(,),M,T,0)
0330 DISP "calculating.."
0340 IF NOT R2 THEN CALL DECOMP(A(,),M,N,S()) @ R2=1
0350 IF S(M)=0 THEN DISP "SINGULAR SYSTEM" @ GOTO 'M'
0360 CALL SOLVE(A(,),X(,),B(,),S(),N)
0370 CALL LIST(X(,),M,1,T,"X")
0380 IF FNK$("solve for new B?","YN")="Y" THEN "S1" ELSE "C"

0390 'I': DIM C(M,N),B(M,1),X(M,1) ! Inverse calculation
0400 DISP "calculating.."
0410 IF NOT R2 THEN CALL DECOMP(A(,),M,N,S()) @ R2=1
0420 IF S(M)=0 THEN DISP "SINGULAR SYSTEM" @ GOTO 'M'
0430 FOR J=1 TO N
0440 FOR I=1 TO M
0450 B(I,1)=0
0460 NEXT I
```

```
0470 B(J,1)=1
0480 CALL SOLVE(A(,),X(,),B(,),S(),N)
0490 FOR I=1 TO M
0500 C(I,J)=X(I,1)
0510 NEXT I
0520 NEXT J
0530 CALL LIST(C(,),M,N,T,"C")
0540 GOTO 'C'

0550 'D': IF T=2 THEN DISP "NOT DONE FOR COMPLEX" @ GOTO 'C' ! det.
0560 DISP "calculating.."
0570 IF NOT R2 THEN CALL DECOMP(A(,),M,N,S()) @ R2=1
0580 D=A(1,1)
0590 FOR I=2 TO M
0600 D=D*A(I,I)
0610 NEXT I
0620 D=S(M)*D
0630 DISP "det= ";D
0640 A$=KEY$ @ IF A$="" THEN 640 ELSE "C" ! display till key hit

0650 'Q': PUT "#38" @ END ! restore blinking prompt and end.
```

MATRIX EDITOR. Allows a matrix of dimension MxN to be edited. The type (T) can be 1 or 2. 1 indicates real, 2 indicates complex. If T=0, then the routine assumes a vector has been passed. The value of T is then changed to the correct type indicator value.

```
0660 SUB EDIT(A(,),M,N,T)
0670 DEF FNK$(D0$,K0$) ! Get which key was hit
0680 DISP D0$
0690 'KEY': K$=KEY$
0700 IF NOT POS(K0$,K$) THEN "KEY"
0710 FNK$=K$
0720 END DEF

0730 DEF FNF$(Y)    ! temporarily change display setting for index vals.
0740 D9$=PEEK$("2F6DC",2) @ STD
0750 FNF$=STR$(Y)
0760 POKE "2F6DC",D9$
0770 END DEF

0780 DEF FND$  ! Create array elemnt prompt
0790 IF NOT (T=1 OR MOD(J,2)) THEN D$=S$ ELSE D$=R$
0800 D1$=D$&"("&FNF$((I+T-1)/T)
0810 IF D$[1,1]<>"B" THEN D1$=D1$&","&FNF$(INT((J+T-1)/T))
0820 IF T=2 AND NOT MOD(J,2) THEN V=-A(I,J) ELSE V=A(I,J)
0830 FND$=D1$&")= "&STR$(V)
0840 END DEF

0850 I=1 @ J=1 @ R=2 @ T$="BRBI"
0860 IF T=2 THEN R$="R" @ S$="I"
0870 IF T=1 THEN R$="A"
0880 IF T=0 THEN R$=T$[1,N] @ S$=T$[3,N+2] @ T=N
```

Figure out which key has been hit and branch to legal choice.

```
0890 'LOP': A$=FNK$(FND$,"Q #38#47#48#50#51") @
                                   IF POS("0134578#",A$) THEN "LOP"
0900 IF LEN(A$)=1 THEN GOSUB UPRC$(CHR$(NUM(A$)+33)) @ GOTO "LOP"
0910 A$[1,1]="K"
0920 GOSUB A$ @ GOTO "LOP"

0930 'K38': INPUT A(I,J) @ IF T=1 THEN GOTO 'K48' ! enter a value
0940 IF MOD(J,2) THEN A(I+1,J+1)=A(I,J) ELSE A(I+1,J-1)=A(I,J) @ A(I,
J)=-A(I,J)

0950 'K48': J=J+1 @ IF J<=N THEN RETURN ! move right
0960 J=1 @ I=I+T @ IF I>M-T+1 THEN I=1
0970 RETURN

0980 'K47': J=J-1 @ IF J THEN RETURN ! move left
0990 J=N @ I=I-T @ IF I<1 THEN I=M-T+1
1000 RETURN

1010 'K50': I=I-T @ IF I>0 THEN RETURN ! move up
1020 I=M-T+1 @ J=J-1 @ IF NOT J THEN J=N
1030 RETURN

1040 'K51': I=I+T @ IF I<=M-T+1 THEN RETURN ! move down
1050 I=1 @ J=J+1 @ IF J>N THEN J=1
1060 RETURN

1070 'A': ON ERROR GOTO 'A' ! user specified move
1080 INPUT "enter ROW,COLUMN";I,J
1090 IF I<1 OR J<1 THEN I=M+1
1100 IF T-1 THEN I=2*I-1 @ J=2*J-1
1110 IF I>M OR J>N THEN DISP "OUT OF BOUNDS" @ GOTO 'A'
1120 OFF ERROR @ RETURN

1130 'R': POP @ SUB END
```

LIST MATRIX SUB PROGRAM. Allows a matrix to be listed by row or column. The array name is passed through parameter B$. The matrix is MxN, and the type (T) is 1 for real values, and 2 for complex values. If a vector is passed, the routine will list by column. A vector is implied by B$ = 'X'.

```
1140 SUB LIST(A(,),M,N,T,B$)

1150 DEF FNF$(Y) ! Create index integer prompts
1160 D9$=PEEK$("2F6DC",2) @ STD
1170 FNF$=STR$(Y)
1180 POKE "2F6DC",D9$
1190 END DEF

1200 DEF FND$  ! Create element prompt
1210 IF B$="X" THEN FND$=FNF$((I+T-1)/T) ELSE
                        FND$=FNF$((I+T-1)/T)&","&FNF$((J+T-1)/T)
1220 END DEF
```

```
1230 D1$=PEEK$("2F946",4) @ DELAY INF,INF !  Temporarily change delay
1240 IF B$="X" THEN GOTO 'C'

1250 'P': DISP "list by Col. or Row?" ! get display choice
1260 A$=KEY$
1270 IF NOT POS("CR",UPRC$(A$[1,1])) THEN 'P'
1280 GOTO UPRC$(A$[1,1])

1290 'C': FOR J=1 TO N STEP T   ! display by column
1300 FOR I=1 TO M STEP T
1310 DISP B$&"("&FND$&")=";A(I,J);
1320 IF T=2 THEN DISP ",";A(I+1,J);"i";
1330 DISP
1340 NEXT I
1350 NEXT J
1360 I=1 @ J=1 @ GOTO 'E'

1370 'R': FOR I=1 TO M STEP T ! display by row
1380 FOR J=1 TO N STEP T
1390 DISP B$&"("&FND$&")=";A(I,J);
1400 IF T=2 THEN DISP ",";A(I+1,J);"i";
1410 DISP
1420 NEXT J
1430 NEXT I
1440 I=1 @ J=1

1450 'E': POKE "2F946",D1$ @ SUB END ! restore delay and exit
```

DECOMPOSITION OF MATRIX. Performs an LU decomposition of an MxN matrix using partial pivoting. The pivoting strategy is recorded in vector S.

```
1460 SUB DECOMP(A(,),M,N,S())
1470 S(M)=1
1480 FOR R0=1 TO M-1
1490 P0=R0
1500 P1=A(R0,R0)

1510 FOR I=R0+1 TO M ! choose largest absolute value for pivot
1520 IF ABS(A(I,R0))>ABS(P1) THEN P0=I @ P1=A(I,R0)
1530 NEXT I
1540 IF A(P0,R0)=0 THEN S(M)=0 @ GOTO 'END' ! quit if singular
1550 S(R0)=P0
1560 IF P0=R0 THEN GOTO 'C'

1570 FOR I=R0 TO N ! row exchange
1580 T=A(R0,I)
1590 A(R0,I)=A(P0,I)
1600 A(P0,I)=T
1610 NEXT I
1620 S(M)=-S(M)

1630 'C': FOR R1=R0+1 TO M ! row r1 <--r1-mult*r0
```

```
1640 M1=A(R1,R0)/A(R0,R0) ! form multiplier
1650 A(R1,R0)=M1 ! and save it.
1660 FOR E=R0+1 TO N
1670 A(R1,E)=A(R1,E)-M1*A(R0,E)
1680 NEXT E
1690 NEXT R1
1700 NEXT R0
1710 IF A(M,N)=0 THEN S(M)=0
1720 'END': SUB END
```

SOLVE ROUTINE. Takes an LU form matrix, pivot strategy vector, B vector, and calculates the X vector for the matrix equation Ax=b.

```
1730 SUB SOLVE(A(,),X(,),B(,),S(),N)
1740 M=N

1750 FOR I=1 TO M-1 ! Permute B and perform reduction
1760 T=B(S(I),1)
1770 B(S(I),1)=B(I,1)
1780 B(I,1)=T
1790 FOR J=I TO M-1
1800 B(J+1,1)=B(J+1,1)-B(I,1)*A(J+1,I)
1810 NEXT J
1820 NEXT I

1830 FOR I=N TO 1 STEP -1 ! back substitution
1840 X(I,1)=B(I,1)
1850 FOR J=I+1 TO M
1860 X(I,1)=X(I,1)-A(I,J)*X(J,1)
1870 NEXT J
1880 X(I,1)=X(I,1)/A(I,I)
1890 NEXT I
1900 SUB END
```

This program calculates a fast Fourier transform from a set of time domain points to a set of frequency domain points. The inverse fast Fourier transform, calculating the set of time domain points from a set of frequency domain points, may also be calculated.

The method used is a modification of the basic FFT algorithm. The modified algorithm takes advantage of the fact that series data is real, and uses the space normally reserved for the imaginary part of the complex sequence to calculate a double-length real transform. This is represented for two "N" length transforms as:

$$Z(n) = X(n) + iY(n) \qquad 0 < n < N \text{ data points}$$

The transform is:

$$Z(m) = X(m) + iY(m)$$

where

$$X(m) = \frac{Z(m) + Z(N-m)^*}{2}$$

$$Y(m) = \frac{Z(m) - Z(N-m)^*}{2i}$$

Z* is the complex conjugate of Z.

The time series F(n) is given by:

$$F(n) = X(2n) + Y(2n+1)$$

The transformation of this is:

$$F(m) = \sum_{n=0}^{N-1} X(2n)w\hat{\ }mn + \sum_{n=0}^{N-1} Y(2n+1)w\hat{\ }mn$$

$$= \sum_{p=0}^{N-1} X(p)w\hat{\ }2mp + \sum_{p=0}^{N-1} Y(p)(w\hat{\ }2mp)(w\hat{\ }m)$$

and:

$$F(m) = X(m) + Y(m)w^{m} \qquad (1)$$

$$F(N-m) = X^{*}(m) - [Y(m)w^{m}]^{*} \qquad (2)$$

Similarly, the inverse transform may be obtained from (1) and (2):

$$Z(m) = \frac{F(m) + F(N-m)^{*}}{2} + iw^{(-m)}\frac{F(m) - F(N-m)^{*}}{2}$$

$$Z(N-m) = \left[\frac{F(m) + F(N-m)^{*}}{2}\right]^{*} - iw^{(-m)}\left[\frac{F(m) - F(N-m)^{*}}{2}\right]^{*}$$

This is simply an interchange of $Z(m)$ and $F(m)$ in (1) and (2), and substitution of $-w^{(-m)}$ for w^{m}.

The advantages gained from this adaptation of the general FFT algorithm for time series data are:

(a) A transform of twice the length can be handled with no increase in storage for input data.

(b) Since the calculation of the transform is structured as an interactive process, intermediate and final results are stored in the same locations used for input.

NOTE:

1. Since $F(0)$ and $F(N)$ are real only, $F(N)$ can be stored in the imaginary location of $F(0)$, i.e., $f(1)$.

2. $w^{m} = c^{(-2im*pi/2N)}$. This is half the minimum value of rotation normally used in an N-point transfer.

3. * denotes the complex conjugate.

REFERENCES

Brigham, E. O., THE FAST FOURIER TRANSFORM, Prentice-Hall, Inc. 1974.

FAST FOURIER TRANSFORM, HP-85 Math Pac, Hewlett-Packard, 1979.

USER INSTRUCTIONS

Comments	Input	Display
1) Run the program.		TIME/FREQ. DATA? (T/F)
2) If time data is to be input, press T and continue with step 3. If frequency data is to be input, go to step 11.	[T]	# OF DATA POINTS?
TIME DOMAIN DATA		
3) Enter number of time domain data points to be used. Must be an integer power of 2 and >2. If it is not, the message "INPUT OUT OF RANGE" will be displayed and you will be prompted to enter the number again. If available memory is insufficient for the specified # of points, the message "NOT ENOUGH MEMORY" will be displayed and the program will again prompt for the number of points.	<N> [ENDLINE]	DATA POINT(nnn)?
4) The program will now prompt for data points 1 through N, where N is the # specified in 3).	<value>[ENDLN] <value>[ENDLN] . . . <value>[ENDLN]	DATA POINT(nnn)? . . . DATA POINT(N)?
5) If any mistakes were made in input, you now have the opportunity to correct the data. Press [Y] to make any changes, or [N] to go to step 6).	[Y]	CHANGES? (Y/N) DATA POINT TO CHANGE?
Enter point # to change.	<nnn> [ENDLINE]	<old value>
Original entry will be displayed for reference.	<new value> [ENDLINE]	CHANGES? (Y/N)
If no (further) changes are necessary, press [N].	[N]	TRANSFORMING...

USER INSTRUCTIONS

Comments	Input	Display
6) Calculation of FFT has begun.		DC TERM =
7) Calculation ends and DC term is displayed. Press any key to continue.	<any key>	MAX FREQ. =
8) Display maximum frequency. Press any key to resume output.	<any key>	FREQ. DOMAIN OUTPUT:
9) Display complex data pairs of calculated frequency domain. First the real, and then the imaginary part of the data pair will be displayed. Press any key to display each value.		1R=<value>
	<any key>	1I=<value>
	<any key>	2R=<value>
	<any key>	.
	.	.
	.	.
	.	NR=<value>
	<any key>	NI=<value>
	<any key>	>
10) For a new problem, go to step 1.		
FREQUENCY DOMAIN DATA		
11) If frequency data is to be input, press [F].	[F]	# OF COEFF. PAIRS?
12) Enter the number of coefficient pairs. This number must be one less than an integer power of 2 (1,3,7,...). If it is not, the message "INPUT OUT OF RANGE" will be displayed and you will be prompted to enter the number again. If available memory is insufficient for the number of pairs specified, the message "NOT ENOUGH MEMORY" will be displayed and the program will again ask for the number of coefficient pairs.	<N> [ENDLINE]	DC TERM=
13) Enter the DC term.	<DC term>	MAX FREQ. TERM=

USER INSTRUCTIONS

Comments	Input	Display
14) Enter the maximum frequency term.	<max freq>	FREQ. DOMAIN DATA -
15) Enter the real and imaginary coefficients for each data pair as prompted.		REAL(1)?
	<real coeff.>	IMAG(1)?
	<imag coeff.>	REAL(2)?
	<real coeff.>	.
	.	.
	.	.
	.	REAL(N)?
	<real coeff.>	IMAG(N)?
	<imag coeff.>	CHANGES? (Y/N)
16) If any mistakes were made in entering the coefficient pairs, they may be corrected by pressing [Y]. If no changes are necessary, press [N] and go to step 17).	[Y]	DATA PAIR TO CHANGE?
Enter # of coefficient pair to correct.	<n>[ENDLINE]	REAL(n)<old entry>
Original entry is displayed for reference.	<new value> [ENDLINE]	IMAG(n)<old entry>
	<new value> [ENDLINE]	CHANGES? (Y/N)
If no (further) changes are necessary, press [N].	[N]	TRANSFORMING...
17) Calculation of inverse FFT has begun.		TIME DOMAIN OUTPUT:
18) Calculation ends and data points are displayed. Press any key to move to next data point.		PT(1)=<value>
	<any key>	PT(2)=<value>
	<any key>	.
	.	.
	.	.
	.	PT(N)=<value>
	<any key>	>
19) For a new problem, go to step 1).		

EXAMPLES

A) For the following set of time domain data points, calculate the Fourier transform to frequency data.

P(1)=1	P(9)=-1
P(2)=1.3Ø66	P(1Ø)=-1.3Ø66
P(3)=1.4142	P(11)=-1.4142
P(4)=1.3Ø66	P(12)=-1.3Ø66
P(5)=1	P(13)=-1
P(6)=.5412	P(14)=-.5412
P(7)=Ø	P(15)=Ø
P(8)=-.5412	P(16)=.5412

USER INSTRUCTIONS

Comments	Input	Display
1) Run the program.		TIME/FREQ. DATA?
2) Choose time data input.	[T]	# OF DATA POINTS?
3) Enter # of points.	16 [ENDLINE]	DATA POINT(1)?
4) Enter point values.	1 [ENDLN]	DATA POINT(2)?
	1.3Ø66 [ENDLN]	DATA POINT(3)?
	1.4142 [ENDLN]	DATA POINT(4)?
	1.3Ø66 [ENDLN]	DATA POINT(5)?
	1 [ENDLN]	DATA POINT(6)?
	.5412 [ENDLN]	DATA POINT(7)?
	Ø [ENDLN]	DATA POINT(8)?
	-.5412 [ENDLN]	DATA POINT(9)?
	-1 [ENDLN]	DATA POINT(1Ø)?
	-1.3Ø66 [ENDLN]	DATA POINT(11)?
	-1.4142 [ENDLN]	DATA POINT(12)?
	-1.3Ø66 [ENDLN]	DATA POINT(13)?
	-1 [ENDLN]	DATA POINT(14)?
	-.5412 [ENDLN]	DATA POINT(15)?
	Ø [ENDLN]	DATA POINT(16)?
	.5412 [ENDLN]	
5) FFT calculation begins.		TRANSFORMING...
6) Frequency domain output.		FREQ DOMAIN OUTPUT:
		DC TERM=Ø
	<any key>	MAX FREQ. =Ø
	<any key>	FREQ DOMAIN OUTPUT:
	<any key>	1R=-1.ØØØØ1ØE+ØØØ
	<any key>	2R= Ø.ØØØØØØE+ØØØ
	<any key>	2I= Ø.ØØØØØØE+ØØØ
	<any key>	3R=-1.339454E-ØØ6
	<any key>	3I=-1.339454E-ØØ6

USER INSTRUCTIONS

Comments	Input	Display
	<any key>	4R= Ø.ØØØØØØE+ØØØ
	<any key>	4I= Ø.ØØØØØØE+ØØØ
	<any key>	5R= 6.134477E-ØØ6
	<any key>	5I=-6.134477E-ØØ6
	<any key>	6R= Ø.ØØØØØØE+ØØØ
	<any key>	6I= Ø.ØØØØØØE+ØØØ
	<any key>	7R=-1.5Ø2234E-ØØ5
	<any key>	7I=-1.5Ø2234E-ØØ5
Done.	<any key>	>

B) For the following set of frequency domain data pairs, perform the inverse Fourier transform to calculate the set of time domain data points. The DC term and the maximum frequency term are Ø.

REAL(1)=1	IMAG(1)=-1
REAL(2)=Ø	IMAG(2)=Ø
REAL(3)=-1.3395E-6	IMAG(3)=-1.3395E-6
REAL(4)=Ø	IMAG(4)=Ø
REAL(5)=6.1345E-6	IMAG(5)=-6.1345E-6
REAL(6)=Ø	IMAG(6)=Ø
REAL(7)=-1.5Ø22E-5	IMAG(7)=-1.5Ø22E-5

USER INSTRUCTIONS

Comments	Input	Display
1) Run the program.		TIME/FREQ. DATA? (T/F)
2) Enter frequency data.	[F]	# OF COEFF. PAIRS?
3) Seven frequency data pairs.	7 [ENDLINE]	DC TERM=
4) DC term is Ø.	Ø [ENDLINE]	MAX FREQ. TERM=
5) Maximum frequency term is Ø.	Ø [ENDLINE]	FREQ. DOMAIN DATA -
6) Begin entry of frequency domain data pairs.		REAL(1)?
	1 [ENDLINE]	IMAG(1)?
	-1 [ENDLINE]	REAL(2)?
	Ø [ENDLINE]	IMAG(2)?
	Ø [ENDLINE]	REAL(3)?
	-1.3395E-6 [ENDLINE]	IMAG(3)?
	-1.3395E-6 [ENDLINE]	REAL(4)?
	Ø [ENDLINE]	IMAG(4)?

USER INSTRUCTIONS

Comments	Input	Display
	Ø [ENDLINE]	REAL(5)?
	6.1345E-6 [ENDLINE]	IMAG(5)?
	-6.1345E-6 [ENDLINE]	REAL(6)?
	Ø [ENDLINE]	IMAG(6)?
	Ø [ENDLINE]	REAL(7)?
	-1.5Ø22E-5 [ENDLINE]	IMAG(7)?
	-1.5Ø22E-5 [ENDLINE]	
7) Inverse FFT calculation begins.		TRANSFORMING...
8) Display time domain data points.		TIME DOMAIN OUTPUT:
		PT(1)= 9.999898E-ØØ1
	<any key>	PT(2)= 1.3Ø6587E+ØØØ
	<any key>	PT(3)= 1.414186E+ØØØ
	<any key>	PT(4)= 1.3Ø6587E+ØØØ
	<any key>	PT(5)= 9.999898E-ØØ1
	<any key>	PT(6)= 5.411945E-ØØ1
	<any key>	PT(7)= 1.ØØØØØØE-Ø12
	<any key>	PT(8)=-5.411945E-ØØ1
	<any key>	PT(9)=-9.999898E-ØØ1
	<any key>	PT(1Ø)=-1.3Ø6587E+ØØØ
	<any key>	PT(11)=-1.414186E+ØØØ
	<any key>	PT(12)=-1.3Ø6587E+ØØØ
	<any key>	PT(13)=-9.999898E-ØØ1
	<any key>	PT(14)=-5.411945E-ØØ1
	<any key>	PT(15)=-1.ØØØØØØE-Ø12
	<any key>	PT(16)= 5.411945E-ØØ1
Done.	<any key>	>

LISTING

```
0010 ! FOURIER TRANSFORM
0020 ! Revision 1.0  4/16/84
0030 !
0040 F9=FLAG(5,FLAG(-10))
0050 OPTION BASE 1 @ OPTION ANGLE RADIANS @ STD @ DELAY 0,0
0060 ON ERROR GOTO 'ERR'
0070 I1$="TF" @ I2$="YN"
0080 O1$="3D,'R=',MZ.6DE" @ O2$="3D,'I=',MZ.6DE"
0090 O3$="'PT(',3D,')=',MZ.6DE"
0100 DISP 'TIME/FREQ. DATA? (T/F)'
0110 'W1': K1$=UPRC$(KEY$) @ IF NOT POS(I1$,K1$) THEN 'W1'
0120 IF K1$='F' THEN SFLAG 1 @ F=-1 ELSE CFLAG 1 @ F=1
```

```
****  Input data points for FFT, or coefficient pairs for   ****
****  inverse FFT. Number of data points must be a power    ****
****  of 2 and greater than 2. Number of coefficient        ****
****  pairs must be one less than a power of 2.             ****
```

```
0130 'IN':
0140 IF NOT FLAG(1) THEN INPUT '# OF DATA POINTS?';N @ GOTO 'INA'
0150 INPUT '# OF COEFF. PAIRS?';N
0160 N=N*2+2
0170 'INA': P=1
0180 IF N=2 THEN 'OR'
0190 FOR L=1 TO 10
0200 P=P*2
0210 IF P=N THEN P1=L @ N2=N/2 @ GOTO 'START'
0220 NEXT L
0230 'OR':
0240 DISP 'INPUT OUT OF RANGE' @ WAIT 1 @ GOTO 'IN'
0250 'START':
0260 DIM R(N2),I(N2)
0270 IF FLAG(1) THEN 'IFFT'
```

```
****  Input time domain data for FFT  ****
```

```
0280 'FFT':
0290 J=0
0300 FOR L=1 TO N2
0310 J=J+1 @ DISP 'DATA POINT(';J;')'; @ INPUT R(L)
0320 J=J+1 @ DISP 'DATA POINT(';J;')'; @ INPUT I(L)
0330 NEXT L
```

```
****  Changes to FFT data  ****
```

```
0340 'C1':
0350 DISP 'CHANGES? (Y/N)'
0360 'W2': K2$=UPRC$(KEY$) @ IF NOT POS(I2$,K2$) THEN 'W2'
0370 IF K2$="N" THEN DISP 'TRANSFORMING...' @ GOTO 'FFTC'
0380 'C2': INPUT 'DATA POINT TO CHANGE?';L
0390 IF L<=0 OR L>2*N2 THEN 'C2'
0400 I2=L/2
0410 IF I2=INT(I2) THEN J=I2 ELSE J=INT(I2)+1
0420 DISP 'DATA POINT(';L;')';
```

LISTING

```
0430 IF J=I2 THEN INPUT '',STR$(I(J));I(J) ELSE INPUT '',STR$(R(J));R(J)
0440 GOTO 'C1'

****  Input DC term, maximum frequency and    ****
****  frequency domain data for inverse FFT.  ****

0450 'IFFT':
0460 INPUT 'DC TERM=';R(1)
0470 INPUT 'MAX FREQ. TERM=';I(1)
0480 DISP 'FREQ. DOMAIN DATA -' @ WAIT 1
0490 FOR L=2 TO N2
0500 DISP 'REAL(';L-1;')'; @ INPUT R(L)
0510 DISP 'IMAG(';L-1;')'; @ INPUT I(L)
0520 NEXT L

****  Changes to inverse FFT data ****

0530 'C3':
0540 DISP 'CHANGES? (Y/N)'
0550 'W3': K3$=UPRC$(KEY$) @ IF NOT POS(I2$,K3$) THEN 'W3'
0560 IF K3$='N' THEN DISP 'TRANSFORMING...' @ GOTO 'IFFTC'
0570 'C4': INPUT 'COEFF. PAIR TO CHANGE?';L
0580 IF L<=0 OR L>N2-1 THEN 'C4'
0590 DISP 'REAL(';L;')'; @ INPUT '',STR$(R(L+1));R(L+1)
0600 DISP 'IMAG(';L;')'; @ INPUT '',STR$(I(L+1));I(L+1)
0610 GOTO 'C3'

****  Start FFT calculation  ****

0620 'FFTC':
0630 K=0
0640 FOR J=1 TO N2-1
0650 L=2
0660 IF K<N2/L THEN 680
0670 K=K-N2/L @ L=L+L @ GOTO 660
0680 K=K+N2/L
0690 IF K<=J THEN 710
0700 A=R(J+1) @ R(J+1)=R(K+1) @ R(K+1)=A @ A=I(J+1) @ I(J+1)=I(K+1)
@ I(K+1)=A
0710 NEXT J
0720 G=.5 @ P2=1
0730 FOR L=1 TO P1-1
0740 G=G+G @ C=1 @ E=0 @ Q=SQR((1-P2)/2)*F
0750 P2=(1-2*(L=1))*SQR((1+P2)/2)
0760 FOR M=1 TO G
0770 FOR J=M TO N2 STEP G+G
0780 K=J+G @ A=C*R(K)+E*I(K) @ B=E*R(K)-C*I(K)
0790 R(K)=R(J)-A @ I(K)=I(J)+B @ R(J)=R(J)+A @ I(J)=I(J)-B
0800 NEXT J
0810 A=E*P2+C*Q @ C=C*P2-E*Q @ E=A
0820 NEXT M
0830 NEXT L
0840 IF FLAG(1) THEN 'IFFTO'
```

```
****  Start inverse FFT calculation  ****

0850 'IFFTC':
0860 A=PI/N2 @ P2=COS(A) @ Q=F*SIN(A)
0870 A=R(1) @ R(1)=A+I(1) @ I(1)=A-I(1)
0880 IF NOT FLAG(1) THEN R(1)=R(1)/2 @ I(1)=I(1)/2
0890 C=F @ E=0
0900 FOR J=2 TO N2/2
0910 A=E*P2+C*Q @ C=C*P2-E*Q @ E=A @ K=N2-J+2 @ A=R(J)+R(K)
0920 B=(I(J)+I(K))*C-(R(J)-R(K))*E @ U=I(J)-I(K)
0930 V=(I(J)+I(K))*E+(R(J)-R(K))*C
0940 R(J)=(A+B)/2 @ I(J)=(U-V)/2 @ R(K)=(A-B)/2 @ I(K)=-(U+V)/2
0950 NEXT J
0960 I(N2/2+1)=-I(N2/2+1)
0970 IF FLAG(1) THEN 'FFTC'
0980 FOR J=1 TO N2
0990 R(J)=R(J)/N2 @ I(J)=I(J)/N2
1000 NEXT J

****  FFT output  ****

1010 'FFTO':
1020 DISP 'DC TERM =';R(1) @ GOSUB 'WAIT'
1030 DISP 'MAX FREQ. =';I(1) @ GOSUB 'WAIT'
1040 DISP 'FREQ DOMAIN OUTPUT:' @ WAIT 1
1050 FOR L=2 TO N2
1060 DISP USING O1$;L-1,R(L) @ GOSUB 'WAIT'
1070 DISP USING O2$;L-1,I(L) @ GOSUB 'WAIT'
1080 NEXT L
1090 GOTO 'DONE'

****  Inverse FFT output  ****

1100 'IFFTO':
1110 DISP 'TIME DOMAIN OUTPUT:' @ WAIT 1
1120 J=1
1130 FOR L=1 TO N2
1140 J=J+1
1150 DISP USING O3$;J-1,R(L) @ GOSUB 'WAIT'
1160 DISP USING O3$;J,I(L) @ GOSUB 'WAIT'
1170 J=J+1
1180 NEXT L
1190 'DONE': F9=FLAG(-10,FLAG(5)) @ PUT '#43' @ END
1200 'WAIT': IF KEY$='' THEN 'WAIT' ELSE RETURN
1210 'ERR': IF ERRL=260 THEN DISP 'NOT ENOUGH MEMORY' @ GOTO 'IN'
1220 DISP ERRM$ @ GOTO 'DONE'
```

Polynomial Root Finder

This program finds all solutions, both real and complex, of $P(x)=0$, where P is a polynomial of the form:

$$P(x)=a(n)x^n + a(n-)x^{(n-1)} + \dots + a(1)x + a(0) = 0$$

Inputs to the program are the degree of the polynomial, the real coeficients a(n)...a(0), tolerances for the evaluation of the function and for each root, and the maximum number of iterations per root.

This program uses Laguerre's method to find the roots of the specified polynomial by computing a sequence of approximations Z(1), Z(2),..., to a root using the formula Z(k+1)=Z(k)+S(k). S(k) is called the Laguerre step, and is defined as:

$$\frac{-nP(Z(k))}{P'(Z(k))+[(n-1)^2(P'(Z(k))^2-n(n-1)P''(z(k))]^{.5}}$$

where P, P', and P'' are the value of the polynomial and its first and second derivatives evaluated at the current iterate k, and n is the degree of the polynomial. The sign in the denominator is chosen to give the Laguerre step of smaller size, which in most cases insures that the roots will be found in order of increasing magnitude.

After an iterate is accepted as a root, synthetic division is used to deflate the polynomial by the factor (x-r) if the root is real, or (x^2-2Re(r)+!!r!!^2) if the root is complex. This saves arithmetic operations, and prevents repetitive convergence to the same root.

For polynomials with only real roots, Laguerre's method will always converge to a root for any choice of real initial estimate. However, for roots of high multiplicity, some loss of accuracy may be observed. If complex roots are present, this method will usually converge to a valid root. If it does not, provisions are made for supplying a new initial estimate and starting the process again.

REFERENCES

Dahlquist, G. and Bjorck, A. NUMERICAL METHODS. Prentice-Hall, 1974.

HP-75 MATH PAC. Hewlett-Packard, 1983

USER INSTRUCTIONS

COMMENTS	INPUT	DISPLAY
1) Run the program.		POLYNOMIAL ROOT FINDER ORDER OF POLYNOMIAL?
2) Input degree of polynomial. Must be a positive integer greater than 1. If it is not, you will be asked to enter it again.	<n>	A(n)=?
3) Enter coefficients of each term, starting with the highest-ordered term.	<a(n)> [ENDLINE] <a(n-1)> [ENDLINE] . . . <a(Ø)> [ENDLINE]	 A(n-1)=? . . . A(Ø)=? TOL. FOR ROOTS=1.E-1Ø
4) Input tolerance for roots. Default value of 1E-1Ø is displayed. If the magnitude of the Laguerre step is less that this value (and 5) is also satisfied) then the current iterate is accepted as a root.	<new value> [ENDLINE]	TOL. FOR FCN=1.E-8
5) Input tolerance for evaluation of function. default value of 1E-8 is displayed. If P(x) for the current iterate is less than this value, and step 4) has been satisfied, the current iterate is accepted as a root.	<new value> [ENDLINE]	MAX # OF ITERATIONS=
6) Input maximum number of iterations for each root. If this number is exceeded before a valid root is found, the message 'NO CONVERGENCE' will be displayed and the user will be allowed to specify a new initial iterate and start the search again.	<nnn> [ENDLINE]	LOOKING FOR ROOTS...

USER INSTRUCTIONS

COMMENTS	INPUT	DISPLAY
7) Calculation of roots has begun. As each root is found, the program will indicate how many roots have been found to this point.		# OF ROOTS FOUND = 1 # OF ROOTS FOUND = 2 . . . # OF ROOTS FOUND = n
		ROOT# 1: R=<value>
8) The real and imaginary parts of each root are displayed. Pressing any key will continue the displaying of the roots.	<any key> <any key> <any key> . . . <any key> <any key>	ROOT# 1: I=<value> ROOT# 2: R=<value> . . . ROOT# N: R=<value> ROOT# N: I=<value> >
9) Done.		

EXAMPLES

A) Find the roots of the polynomial given below. Use default values for the tolerances and limit iterations to 1Ø.

P(x) = 5x^6-45x^5+225x^4-425x^3+17Øx^2+37Øx-5ØØ

USER INSTRUCTIONS

COMMENTS	INPUT	DISPLAY
1) Run the program.		POLYNOMIAL ROOT FINDER ORDER OF POLYNOMIAL?
2) Enter order of polynomial.	6 [ENDLINE]	A(6)=?
3) Enter the coefficients, starting with the highest order term.	5 [ENDLINE]	A(5)=?
	-45 [ENDLINE]	A(4)=?
	225 [ENDLINE]	A(3)=?
	-425 [ENDLINE]	A(2)=?
	17Ø [ENDLINE]	A(1)=?
	37Ø [ENDLINE]	A(Ø)=?
	-5ØØ [ENDLINE]	TOL. FOR ROOTS=1.E-1Ø
4) Use default value.	[ENDLINE]	TOL. FOR FCN=1.E-8
5) Use default value.	[ENDLINE]	MAX # OF ITERATIONS=
6) Limit to 1Ø iterations.	1Ø [ENDLINE]	LOOKING FOR ROOTS...
7) Calculation of roots begins.		# OF ROOTS FOUND = 2 # OF ROOTS FOUND = 3 # OF ROOTS FOUND = 4 # OF ROOTS FOUND = 6
		ROOT# 1: R= 1.ØØØØØØE+ØØØ
8) Display real and imaginary parts of each root.	<any key>	ROOT# 1: I= 1.ØØØØØØE+ØØØ
	<any key>	ROOT# 2: R= 1.ØØØØØØE+ØØØ
	<any key>	ROOT# 2: I=-1.ØØØØØØE+ØØØ
	<any key>	ROOT# 3: R=-1.ØØØØØØE+ØØØ
	<any key>	ROOT# 3: I= Ø.ØØØØØØE+ØØØ
	<any key>	ROOT# 4: R= 2.ØØØØØØE+ØØØ
	<any key>	ROOT# 4: I= Ø.ØØØØØØE+ØØØ
	<any key>	ROOT# 5: R= 3.ØØØØØØE+ØØØ
	<any key>	ROOT# 5: I=-4.ØØØØØØE+ØØØ
	<any key>	ROOT# 6: R= 3.ØØØØØØE+ØØØ
	<any key>	ROOT# 6: I= 4.ØØØØØØE+ØØØ
	<any key>	>
9) Done.		

LISTING

```
ØØ1Ø DEFAULT ON
ØØ2Ø ! POLYNOMIAL ROOT FINDER
ØØ3Ø ! Revision 1.ØØ  4/8/84
ØØ4Ø STD @ OPTION BASE Ø @ INTEGER I,J,K,N
ØØ5Ø AØ$='YN' @ A7$='R' @ A8$='I'
ØØ6Ø A9$="'ROOT#',DD,': ',A,'=',MZ.6DE"
ØØ7Ø DEF FNF2(U1,V1)=SQRT(U1*U1+V1*V1)
ØØ8Ø DISP "POLYNOMIAL ROOT FINDER" @ WAIT .5

****  INPUT ORDER, COEFFICIENTS, TOLERANCES   ****
****  AND MAXIMUM NUMBER OF ITERATIONS        ****

ØØ9Ø 'ORD':
Ø1ØØ INPUT "ORDER OF POLYNOMIAL? ";N
Ø11Ø IF N<=1 OR N#IP(N) THEN DISP "INVALID ORDER" @ GOTO 'ORD'
Ø12Ø ON ERROR GOTO 'MEM'
Ø13Ø DIM AØ(N),RØ(N,2),L(2),C9(N)
Ø14Ø FOR I=N TO Ø STEP -1
Ø15Ø DISP 'A(';STR$(I);') =';  @ INPUT AØ(I)
Ø16Ø NEXT I
Ø17Ø 'TOL':
Ø18Ø E1=.ØØØØØØØØØ1 @ SCI Ø
Ø19Ø INPUT 'TOL. FOR ROOTS=',STR$(E1);E1
Ø2ØØ INPUT 'TOL. FOR FCN=',STR$(E1*1ØØ);E2
Ø21Ø STD
Ø22Ø 'IT': INPUT 'MAX # OF ITERATIONS=';IØ
Ø23Ø IF IP(IØ)#IØ OR NOT IØ THEN 'IT'
Ø24Ø DISP 'LOOKING FOR ROOTS...' @ WAIT .5
Ø25Ø J=N @ K=Ø  @ X=Ø @ Y=Ø
Ø26Ø 'FINDR':

****  IS ZERO A ROOT?  ****

Ø27Ø IF NOT AØ(Ø) THEN 'FAR'

****  INPUT NEW GUESS IF ITERATION LIMIT EXCEEDED  ****

Ø28Ø 'LOOP': K=K+1
Ø29Ø IF K>IØ THEN DISP 'NO CONVERGENCE' @ WAIT 1 @ GOSUB 'NUROOT'

****  CALCULATE P, P', AND P'' AT Z(x,y)  ****

Ø3ØØ R=X*X+Y*Y @ D=X+X
Ø31Ø DØ=Ø @ D1=Ø @ CØ=Ø @ C1=Ø
Ø32Ø B1=AØ(J) @ BØ=AØ(J-1)+D*AØ(J)
Ø33Ø FOR I=J-2 TO Ø STEP -1
Ø34Ø IF I=Ø THEN D=X
Ø35Ø IF I>J-4 THEN 38Ø
Ø36Ø V=D1*R @ D1=DØ
Ø37Ø DØ=C1+D*DØ-V
Ø38Ø V=C1*R @ C1=CØ
Ø39Ø CØ=B1+D*CØ-V
Ø4ØØ V=B1*R @ B1=BØ
Ø41Ø BØ=AØ(I)+D*BØ-V
Ø42Ø NEXT I
```

LISTING

```
****  BEGIN CALCULATION OF LAGUERRE STEP  ****

0430 P(1,1)=B0 @ P(1,2)=B1*Y
0440 P(2,1)=B1-2*Y*Y*C1 @ P(2,2)=2*Y*C0
0450 P(3,1)=2*C0-8*Y*Y*D0 @ P(3,2)=2*Y*(3*C1-4*Y*Y*D1)
0460 CALL MULT(P(1,1),P(1,2),P(3,1),P(3,2),S1,S2)
0470 S1=-S1*J*(J-1) @ S2=-S2*J*(J-1)
0480 CALL MULT(P(2,1),P(2,2),P(2,1),P(2,2),S3,S4)
0490 S3=S3*(J-1)^2 @ S4=S4*(J-1)^2
0500 CALL ADD(S3,S4,S1,S2,S1,S2)
0510 CALL SQAROOT(S1,S2,S1,S2)
0520 CALL ADD(P(2,1),P(2,2),S1,S2,L1,L2)
0530 CALL ADD(P(2,1),P(2,2),-S1,-S2,L3,L4)
0540 CALL DIVID(P(1,1),P(1,2),L1,L2,L1,L2)
0550 CALL DIVID(P(1,1),P(1,2),L3,L4,L3,L4)

****  CHOOSE SIGN OF DENOMINATOR TO  ****
****  PRODUCE SMALLER LAGUERRE STEP  ****

0560 IF FNF2(L1,L2)<FNF2(L3,L4) THEN S1=L1 @ S2=L2 ELSE S1=L3 @ S2=L4
0570 L(1)=-J*S1 @ L(2)=-J*S2

****  FORCE REAL ROOT  ****

0580 IF ABS(L(2))<.0001*FNF2(L(1),L(2)) THEN L(2)=0 @ Y=0
0590 X=X+L(1) @ Y=Y+L(2)

****  CHECK FOR VALID ROOT  ****

0600 IF FNF2(L(1),L(2))>ABS(E1) THEN 'LOOP'
0610 IF FNF2(P(1,1),P(1,2))<ABS(E2) THEN 'FAR'
0620 DISP 'INVALID ROOT FOUND' @ WAIT 1 @ GOSUB 'NUROOT' @ GOTO 'LOOP'

****  FOUND VALID ROOT(S) - IF COMPLEX  ****
****  ROOT ASSUME CONJUGATE IS A ROOT   ****

0630 'FAR':
0640 R0(J,1)=X @ R0(J,2)=Y @ IF Y THEN GOSUB 'ROOTI'
0650 DISP '# OF ROOTS FOUND =';N-J+1
0660 IF NOT FNF2(R0(J,1),R0(J,2)) THEN GOSUB 'ROOT0' ELSE GOSUB 'DEFLATE'
0670 J=J-1 @ K=0 @ IF NOT J THEN 'DR'
0680 IF J=1 THEN R0(1,1)=-A0(0)/A0(1) ELSE K=0 @ X=0 @ Y=0 @ GOTO 'FINDR'

****  DISPLAY ROOTS  ****

0690 DISP '# OF ROOTS FOUND =';N
0700 'DR':
0710 FOR I=N TO 1 STEP -1
0720 DISP USING A9$;N-I+1,A7$,R0(I,1) @ GOSUB 'WAIT'
0730 DISP USING A9$;N-I+1,A8$,R0(I,2) @ GOSUB 'WAIT'
0740 NEXT I
0750 'DONE': PUT "#43" @ END
0760 'WAIT': IF KEY$='' THEN 'WAIT' ELSE RETURN
```

```
****  DEFLATION ROUTINES - IF ROOT IS REAL, DEFLATE BY   ****
****  LINEAR FACTOR (x-r). IF ROOT IS COMPLEX, DEFLATE   ****
****  BY BINOMIAL X^2-2Re(r)X+!!r!!^2.                   ****

0770 'DEFLATE':
0780 IF Y THEN 'DEFLATEI'
0790 C9(J-1)=A0(J)
0800 FOR I=J-1 TO 1 STEP -1
0810 C9(I-1)=A0(I)+C9(I)*R0(J,1)
0820 NEXT I
0830 FOR I=0 TO J-1 @ A0(I)=C9(I) @ NEXT I
0840 RETURN
0850 'DEFLATEI':
0860 C9(J-1)=A0(J+1) @ C9(J)=0 @ IF J=1 THEN RETURN
0870 FOR I=J-1 TO 1 STEP -1
0880 C9(I-1)=A0(I+1)+C9(I)*R0(J,1)*2-(R0(J,1)^2+R0(J,2)^2)*C9(I+1)
0890 NEXT I
0900 FOR I=0 TO J-1 @ A0(I)=C9(I) @ NEXT I
0910 RETURN

****  DEFLATE FOR ZERO ROOT  ****

0920 'ROOT0':
0930 FOR I=0 TO J-1 @ A0(I)=A0(I+1) @ NEXT I
0940 RETURN

****  SET NEXT ROOT TO COMPLEX CONJUGATE  ****
****  OF COMPLEX ROOT JUST FOUND          ****

0950 'ROOTI':
0960 J=J-1 @ R0(J,1)=X @ R0(J,2)=-Y
0970 RETURN

****  ASK FOR NEW GUESS IF FAILURE TO CONVERGE  ****

0980 'NUROOT':
0990 DISP 'NEW GUESS? (Y/N)'
1000 'W0': A1$=KEY$ @ IF NOT POS(A0$,UPRC$(A1$)) THEN 'W0'
1010 IF UPRC$(A1$)='N' THEN 'DONE'
1020 INPUT 'NEW u =',STR$(X);X
1030 INPUT 'NEW v =',STR$(Y);Y
1040 K=0
1050 RETURN
1060 'MEM':
1070 IF ERRN#24 THEN DISP ERRM$ @ GOTO 'DONE'
1080 DISP 'LOW MEM - REDUCE ORDER' @ WAIT 1 @ GOTO 'ORD'

****  ADDITION OF COMPLEX NUMBERS  ****

1090 SUB ADD(U1,V1,U2,V2,U,V)
1100 U=U1+U2 @ V=V1+V2
1110 END SUB
```

LISTING

**** MULTIPLICATION OF COMPLEX NUMBERS ****

```
1120 SUB MULT(U1,V1,U2,V2,U,V)
1130 U=U1*U2-V1*V2 @ V=U1*V2+U2*V1
1140 END SUB
```

**** DIVISION OF COMPLEX NUMBERS ****

```
1150 SUB DIVID(U1,V1,U2,V2,U,V)
1160 CALL MULT(U1,V1,U2,-V2,Z1,Z2)
1170 D5=U2*U2+V2*V2 @ IF NOT D5 THEN U=0 @ V=0 @ GOTO 1200
1180 U=Z1/D5
1190 V=Z2/D5
1200 END SUB
```

**** SQUARE ROOT OF A COMPLEX NUMBER ****

```
1210 SUB SQAROOT(U1,V1,U,V)
1220 A2=SQR((SQR(U1*U1+V1*V1)+ABS(U1))/2)
1230 IF NOT A2 THEN U=0 @ V=0 @ GOTO 1280
1240 B2=V1/(2*A2)
1250 IF U1>=0 THEN U=A2 @ V=B2 @ GOTO 1280
1260 U=ABS(B2)
1270 IF B2>=0 THEN V=A2 ELSE V=-A2
1280 END SUB
```

NOTES

HEWLETT PACKARD

00071-90064

Printed in USA

5/16/84

ERRATA: HP-71 MATH SOLUTIONS BOOK

Page 57:

Instructions	Input	Display
3) Enter the coefficients, starting with the highest order term.	5[ENDLINE]	A(5)=?
	-45[ENDLINE]	A(4)=?
	225[ENDLINE]	A(3)=?
	-425[ENDLINE]	A(2)=?
	170[ENDLINE]	A(1)=?
	370[ENDLINE]	A(0)=?
	-500[ENDLINE]	TOL. FOR ROOTS=1.E-10
7) Calculation of roots begins.		# OF ROOTS FOUND= 2
		# OF ROOTS FOUND= 3
		# OF ROOTS FOUND= 4
		# OF ROOTS FOUND= 6
8) Display real and imaginary parts of each root.		ROOT# 1: R= 1.000000E+000
	<any key>	ROOT# 1: I= 1.000000E+000
	<any key>	ROOT# 2: R= 1.000000E+000
	<any key>	ROOT# 2: I=-1.000000E+000
	<any key>	ROOT# 3: R=-1.000000E+000
	<any key>	ROOT# 3: I= 0.000000E+000

Page 58:

```
0250 J=N @ K=0 @ X=0 @ Y=0
0270 IF NOT A0(0) THEN 'FAR'
```

Page 59:

```
0720 DISP USING A9$;N-I+1,A7$,R0(I,1) @ GOSUB 'WAIT'
0730 DISP USING A9$ N-I+1,A8$,R0(I,2) @ GOSUB 'WAIT'
```

Page 61:

```
1220 A2=SQR((SQR(U1*U1+V1*V1)+ABS(U1))/2)
```

5/16/84

ERRATA: HP-71 MATH SOLUTIONS BOOK

Page 9:

```
0200 IF NOT FLAG(1) EXOR FLAG(2) THEN DISP 'I/O INCORRECT' @ WAIT 2 ELSE 'A1
0510 IF FLAG (2) THEN CALL C2R(X0,Y0)
```

Page 15:

4) Key in the number of points.	9 [ENDLINE]	Interval length=0
5) Key in the interval length for the partitions.	.25[ENDLINE]	E(0)=0

Page 18:

```
0580 R=A(0)+A(N)
```

Page 27:

```
0090 M$=KEY$ @ IF M$="" THEN 90
0160 A$=KEY$ @ IF A$="" THEN 160
0230 A$=KEY$ @ IF A$="" THEN 230 ELSE "F"
```

Page 28:

```
0410 A$=KEY$ @ IF A$="" THEN 410
```

Page 34:

?

3) Key in the number of rows.	5 [ENDLINE]	A(1,1)= 0.0000

Page 36:

3) The order is 2.	2 [ENDLINE]	R(1,1)= 0.0000
8) Choose solve option.	S	BR(1)= 4.0000

Page 48:

?

6) Frequency domain output.		DC TERM=0
	<any key>	MAX FREQ. =0
	<any key>	FREQ DOMAIN OUTPUT:
		1R= 1.000010E+000